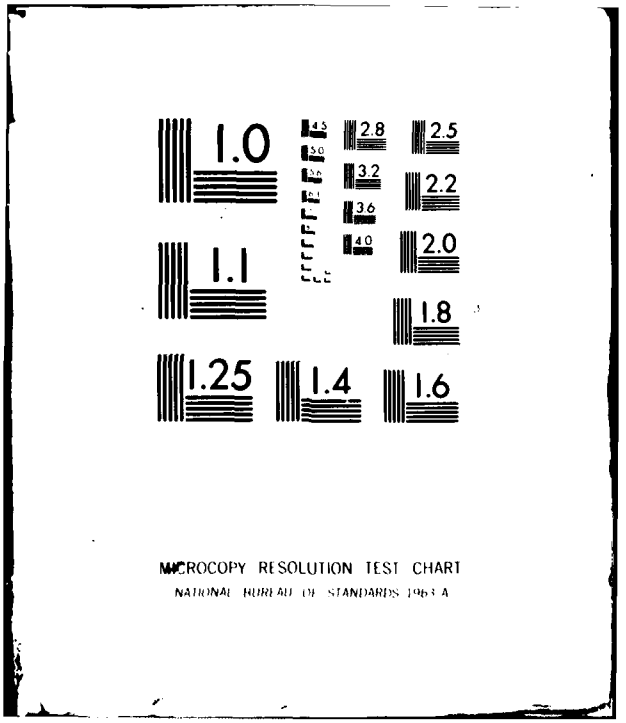
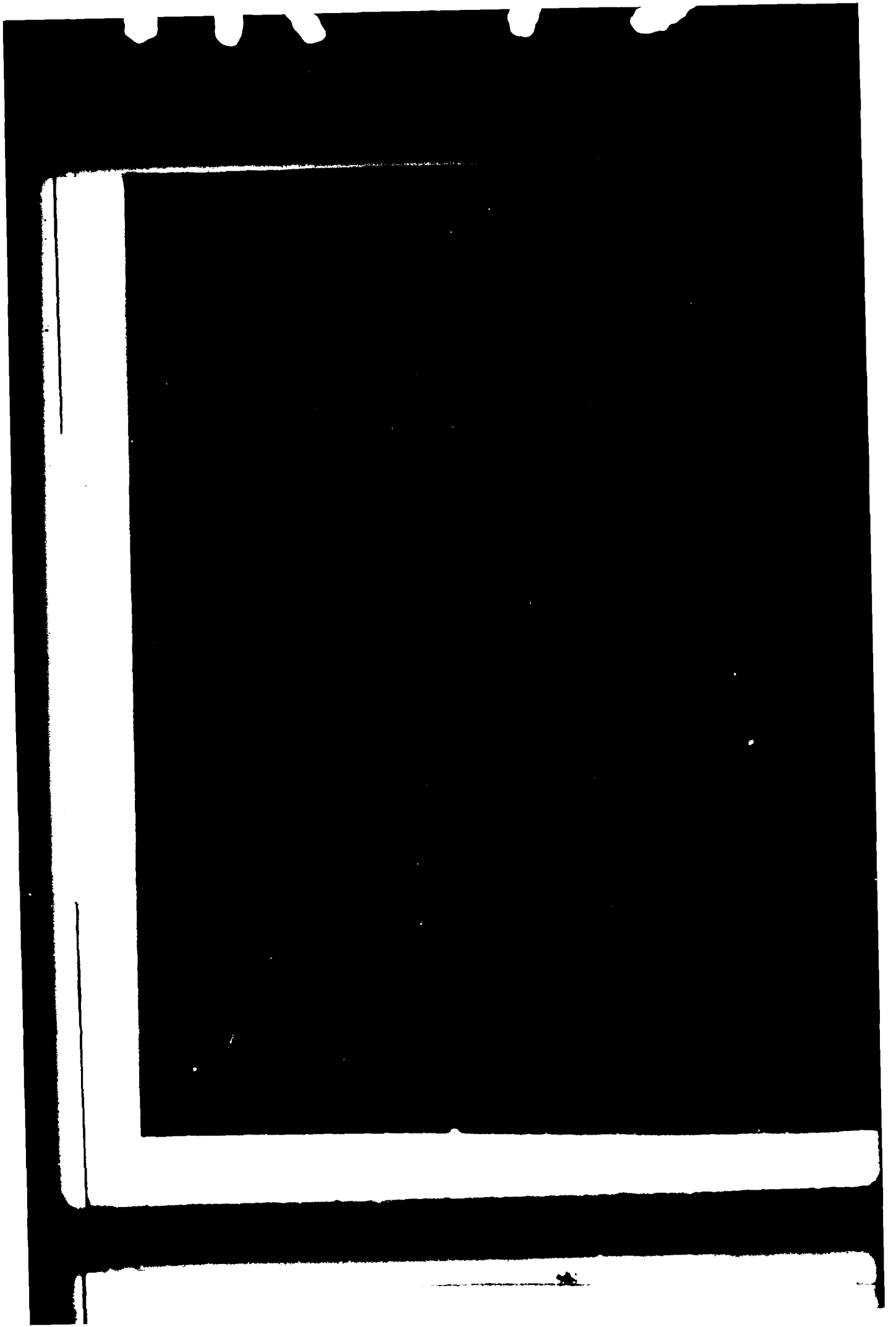
AD-A083 991 AIR FORCE AVIONICS LAB WRIGHT-PATTERSON AFB OH F/G 9/2
LOG POLAR QUANTIZATION.(U)
OCT 79 R T COKER, G D COUTURIER
UNCLASSIFIED AFAL-TR-79-1075 NL


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0
MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS 1963 A



SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: AFAL-TR-79-1075	2. GOVT ACCESSION NO.: AD-A083 991
3. RECIPIENT'S CATALOG NUMBER:	
4. TITLE *(and Subtitle)*: LOG POLAR QUANTIZATION.	5. TYPE OF REPORT & PERIOD COVERED: Interim Technical Report. 1 Apr 77 — 1 Sep 78
	6. PERFORMING ORG. REPORT NUMBER:
7. AUTHOR(s): Robert T. Coker, Guy D. Couturier	8. CONTRACT OR GRANT NUMBER(s):
9. PERFORMING ORGANIZATION NAME AND ADDRESS: Air Force Avionics Laboratory (DHE-1), AF Wright Aeronautical Laboratories, AFSC, Wright-Patterson Air Force Base, Ohio 45433	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: Program Element Project 6096 Task 609831 Work Unit 60963101
11. CONTROLLING OFFICE NAME AND ADDRESS: Air Force Avionics Laboratory (DH), AF Wright Aeronautical Laboratories, AFSC, Wright-Patterson Air Force Base, Ohio 45433	12. REPORT DATE: Oct 79
	13. NUMBER OF PAGES: 124
14. MONITORING AGENCY NAME & ADDRESS *(if different from Controlling Office)*: 124	15. SECURITY CLASS. *(of this report)*: Unclassified
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT *(of this Report)*

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT *(of the abstract entered in Block 20, if different from Report)*

18. SUPPLEMENTARY NOTES

19. KEY WORDS *(Continue on reverse side if necessary and identify by block number)*

Signal Processing
Quantization
Polar Coordinates
Register Simulation
Gate Simulation

20. ABSTRACT *(Continue on reverse side if necessary and identify by block number)*

Log Polar Quantization was developed as an alternative to real-imaginary quantization for complex digital signal processing. The motivation was to achieve a better match between the quantization scheme and complex signal processing characteristics, without precluding the effective use of microprocessor or bit slice processor architectures. The subject report provides information on a specific Log Polar Vector Adder which was breadboarded and simulated, to provide an example of the implementation of such a quantization scheme.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

011670

FOREWORD

This report describes an in-house effort conducted at the Air Force Avionics Laboratory, Electronic Technology Division, Microelectronics Branch, Wright-Patterson AFB, Ohio, under USAF Project 6096 entitled, "Microelectronic Technology," Task 31, entitled, "High Speed Signal Processing Technology," and work unit 01, entitled, "Signal Processing Investigations."

The work reported herein was performed by the authors during the period 1 Apr 77 to 1 Sep 78, under the direction of Mr. Robert T. Coker (AFAL/DHE-1). The report was released by the authors in Dec 78.

The authors gratefully acknowledge the contributions of Mr. Ben Carroll, AVCO, who spent many hours laying out and building the breadboard hardware (Section III.B).

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

SECTION I

INTRODUCTION

Digital signal processing is constantly moving into areas occupied by analog signal processing. This trend is basically paced by the development of faster and denser integrated circuits. Although real-time complex digital signal processing has been around for some time, most implementations have been an extension of real signal processing hardware and techniques.

An imaginary channel is added and the processing is performed in the real/imaginary coordinated system. This coordinated system is basically mismatched to the operational and quantization requirements of complex signal processing. Complex signal processing primarily deals with phase-to-amplitude and amplitude-to-phase information transforms, hard limiting of amplitude, storage of phase only information, etc. The real/imaginary coordinate system does not provide for the efficient separation of phase and amplitude information. Also, either excessive bits or floating point hardware is required to handle large amplitude dynamic ranges.

This mismatch is further detailed in Section II which covers the conception of a Log Polar Quantization scheme and its key element, the Log Polar Vector Adder.

Section III discusses the implementation of the Vector Adder in a register level simulation, a hardware breadboard, and a gate level simulation.

Section IV contains the conclusion.

SECTION II

THE CONCEPT

This section discussed the Log Polar quantization and the design of a Log Polar Vector Adder circuit. The format this section basically follows is the pattern of the idea's conception. The basic decision criteria used were low cost and maximum compatability with commercial microprocessor designs. Also, the implementation of the Vector Adder was based more on practicality than on optimization analysis.

For digital signal processing, complex data is typically quantized as linear real and imaginary word pairs of 8 to 16 bits each. This quantization technique is an extension of real data quantization and is a poor match to the phase and amplitude operations typical of signal processing. Amplitude weighing requires two multiplications on two words even though it is fundamentally a one-multiply operation. Phase rotation requires a sine table lookup and four multiplications, even though it is fundamentally an add operation. Amplitude detection requires the square root of the sum of the squares of the real and imaginary words or an approximation thereof, even though fundamentally no operation should be required. Signal processing frequently involves conversions between amplitude and phase information, and the storage of phase only or amplitude only information. Complex signal processing typically interfaces the man-computer to the receiver-transmitter; the first prefers amplitude information, and the latter prefers phase information. The real and imaginary quantization technique requires the storage of both phase and amplitude information to equivalent precisions even if only one is required. Also, real and imaginary quantization typically requires two matched mixer-IF-A/D channels or an over sampled real channel and some extra processing.

Polar quantization is a better fit, because the amplitude and phase information are independently quantized. No processing is required to interface to the amplitude world of the man-computer. The transmitter-receiver interface is equally straightforward, but admittedly less familiar and developed. No phase-to-digital converters or polar-phase

modulators are known to exist. However, there are no fundamental or practical constraints to their implementation. In fact, they could provide considerable simplifications in the R.F.-to-digital conversion hardware.

Unfortunately signal processing requires a vector addition function. The typical way to implement this function for polar quantization, is to convert to real and imaginary quantization. Needless to say, the cure is worse than the disease. A direct table lookup operation is also generally impractical because of the table size. If one of the vectors can be normalized with respect to the other, the table size can be reduced by the square root. The phase normalization/inverse-normalization is a subtraction/addition process; however, the amplitude process requires a division/multiplication process. The cure is better, but still worse than the disease. If the amplitude is quantized on a logarithmic scale the normalization becomes a subtraction/addition process. Also, the logarithmic amplitude scale is better suited to complex signal processing. Large dynamic ranges can be stored in a limited number of bits without the hardware complexity of floating point, and the quantization accuracy is a percent of value ($\pm$xdb) rather than plus or minus the least significant bit. If the logarithmic quantization is based on integer powers of two (computers like powers of two) the large dynamic range is achieved in a few bits, but the accuracy is only plus or minus 6.0 db. Note the amplitude values of Table 1 for the four most significant bits. This accuracy is not very useful, because of the false signals (quantization noise) caused by the amplitude quantization errors. Floating point is the typical solution, but its mixed quantization scheme requires an unacceptable hardware penalty in this case. The solution is to extend the logarithmic scale downward with fractional powers of 2. Note the amplitude values of Table 1 for the three least significant bits. Negative amplitude values are not included in the quantization; negative values are handled by 180° phase reversals (changing the most significant bit of phase). Also note that zero is not included. The smallest value is a ratio of one. Don't panic, zero amplitude signals do not exist in the real world. Complex signal amplitudes are ratios or referenced values; signal to noise, dbm, dbw, etc.

TABLE 1

AMPLITUDE AND PHASE SCALES

AMPLITUDE			PHASE	
OCTAL	db	RATIO	OCTAL	DEGREES
0	0.00	1.0000	0	0.00
1	0.75	1.0905	1	2.81
2	1.50	1.1892	2	5.63
3	2.25	1.2968	3	8.44
4	3.00	1.4142	4	11.25
5	3.75	1.5422	5	14.06
6	4.50	1.6818	6	16.88
7	5.25	1.8340	7	19.69
10	6.00	2.0000	10	22.50
20	12.00	4.0000	20	45.00
30	18.00	8.0000	30	67.50
40	24.00	16.0000	40	90.00
100	48.00	256.0000	100	180.00
140	72.00	4096.0000	140	270.00
177	95.25	60097.0000	177	357.20

We have now reached the point of plausibility, but we still have to reach the point of practicality. The main problem left is the peak and integrated values of the quantization noise. This will be the main limitation on applications of this Log Polar implementation. The quantization noise will correlate to produce false signals. It is desirable to reduce this peak quantization noise toward the level of the average quantization noise. This is accomplished in the vector adder by the addition of pseudo-random amplitude and phase noise at half the value of the least significant bit of both phase and amplitude. This has no practical effect on real world signals, because they are not aligned to the quantization scale. This noise technique comes virtually free by offsetting the calculated table values by half the value of the least significant bit, and by adding random values to the least significant bit of the normalized phase and amplitude. The phase is randomized based on the least significant bit of the normalized amplitude and the amplitude on the least significant bit of the normalized phase. This provides an acceptable randomization, while keeping a deterministic output that is a function of input. This is important to the testability of the circuit.

A specific quantization scale is picked next and a practical vector adder circuit is designed. No attempt was made to define an optimum scale, only a practical one for demonstration purposes. Before we can proceed, we need to pause and try to visualize the total signal processing subsystem. Note that once we have the vector adder circuit all other complex signal processing functions are additions and subtractions. Also, the table size of the vector adder grows approximately exponentially in proportion to the word size.

We would certainly like to take advantage of the explosion in commercial microprocessors. The main word size of these processors is currently 8 bits and will be 16 bits in the near future. Thus, eight bits for amplitude and eight bits for phase looks like a good choice so far. But, a better choice might be seven bits of amplitude and seven bits of phase. This leaves a one bit buffer between the phase and amplitude so that a single 16 bit Arithmetic Logic Unit can do separate arithmetic operations, on both seven bit fields, simultaneously. The

seven bit phase field is stored in the least significant bits (1 through 7). The eighth bit is a buffer. The seven bit amplitude field is stored in the next seven bits (8 through 15). The sixteenth bit (sign bit) is used for amplitude overflow. The seven bits of phase provide (360°/128) $\pm$2.8° accuracy. This accuracy is not outstanding but is adequate for many applications. It is also a good match to the phase accuracy of most low-to-moderate-cost receiver/transmitter systems. The seven bits of amplitude provide almost 96 db of dynamic range with $\pm$0.75 db accuracy (Table 1). Again this is a good match to most low to moderate cost receiver/transmitter and display systems. The main question now is the limitations of the quantization noise for the chosen scale.

The quantization noise has not been analytically quantified, because the logarithmic amplitude quantization makes the vector addition a two-dimensional, nonlinear process. However, the Fast Forrier Transform (FFT) has been simulated and the peak quantization noise was more than 38 db below signal level. The pseudo-random noise addition keeps the peak quantization noise very near the integrated level. The intermodulation products were more than 30 db down and a weighted FFT with minus 30 db sidelobes was obtained. Better performance can be achieved from more bits, but the vector add table expands very rapidly.

A block diagram of the vector adder is shown in Figure 1. The circuit is not clocked. It consists of combinational logic. Internal registers could be added to pipeline the logic and increase its throughput. A final chip design would include input and/or output latches. Also, controlled inversion of the most significant bit of phase would be included for vector subtractions. This only requires the addition of two input lines and two exclusive Or gates. Two fourteen bit vectors "VI" and "V2" are input at the top of the figure. The "A" and "P" designate the amplitude and phase terms, respectively. The fourteen bit output is at the bottom. "AO" is the seven bit amplitude field, and "PO" is the seven bit phase field. Amplitude operations occur on the left side of the figure and phase operations on the right.

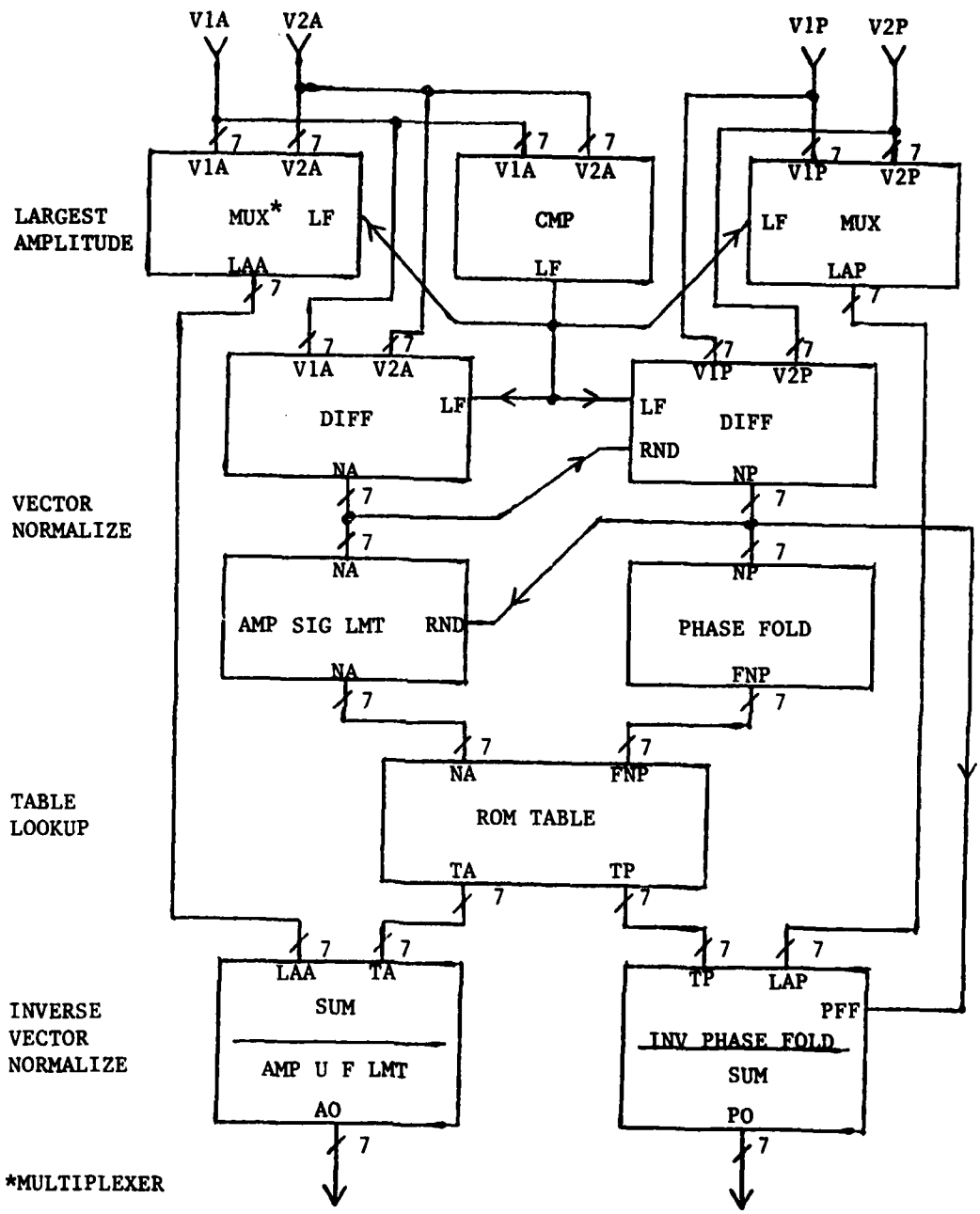


Figure 1. Vector Adder Block Diagram

The circuit function is reasonably straightforward, if the basic processes are kept in mind. The four basic processes are select largest amplitude, vector normalize, table lookup, and inverse vector normalize. All other functions of the circuit either reduce the hardware or enhance the overall function. The first basic process "Largest Amp" determines the input vector with the largest amplitude. This is done in the "CMP" block. Also, the largest vector is selected in the two "MUX" (multiplexer) blocks. This selection is determined by the Largest Flag ("LF"). The second basic process "Vector Normalize" is performed next.

The smallest amplitude vector is normalized with respect to the largest in the two "DIFF" blocks. "LF" causes the smallest vector to be subtracted from the largest. The result is a positive amplitude number equal to the absolute value of the smallest minus the largest. Zero (Ø) represents a normalize amplitude of one (0 db). Eight represents -3 db and so on. This is done because table addresses are easier to implement as positive numbers. Subtracting the smallest from the largest also results in negative values of normalize phase. This is corrected in the table by using negative phase values for the positive phase input values. This phase inversion could be removed in future versions, but care should be taken or no significant circuit reductions will result. The "RND" input to the phase "DIFF" block is its carry input and is from the least significant bit of the normalized amplitude. Remember, the calculated table values are offset by half the value of the least significant bit, so that a random choice is made between $\pm 1/2$ least significant bit.

Because of the large dynamic range and limited accuracy of the amplitude quantization, the vector addition is not significant for large amplitude differences. In these cases the change in amplitude of the largest vector is less than the least significant bit of quantization. The "Amp Sig Lmt" block detects this condition to reduce the required size of the Read Only Memory (ROM) table. Prior to the application of this limit the least significant bit of the normalized phase is added at the "RND" input to rendomize the least significant bit of the normalized amplitude. Because the normalized vector addition is symmetrical about the 0° to 180° phase line, the phase is folded into the upper

hemisphere (0° to 180°) if it is between 180° and 360° (Figure 2.) This is performed in the "Phase Fold" block and reduces ROM table size requirements by a factor of two.

The next basic process is "Table Lookup," which is performed in the block labeled "ROM Table." The table vectorially adds the normalized, amplitude limited, and phase folded vector to a unit vector at 0° phase.

Both the amplitude and phase output fields are offset by minus one to reduce the table size. These offsets are corrected by adding one in both the amplitude and phase inverse normalize blocks.

The last basic process is the "Inverse Vector Normalize." The amplitude field of the vector out of the table is added to the amplitude field of the largest vector, which was selected in the first basic process. Additional logic "Amp U F LMT" prevents an underflow condition by setting the amplitude field to zero when an underflow condition is detected. Thus, the user need not be concerned with amplitude underflow errors generating false signals. Amplitude overflow errors are not possible because the ROM table is actually implemented as a vector addition divided by two. Thus, the amplitude output can never be larger than the largest amplitude input. See Figure 3. This not only simplifies the hardware implementation, but frees the user from being concerned with false signals caused by amplitude saturation. FFT's implemented with this "Normalize" vector addition are equivalent to fixed scaling (dividing by two) in each FFT stage. The noise drops out from under the signal, instead of the signal growing out of the noise. The FFT can be scaled up to saturation at any stage without concern for overflows or allowing extra bits for signal growth. The division by two can be simply undone by adding 8 (3 db) to the amplitude field. Also, the amplitude can be limited (clipped) to any level at any time without affecting the stored phase information.

The phase field of the table output is unfolded according to the Phase Foldover Flag ("PPF"). This is done in the "Inv Phase Fold" block. The phase field of the previously saved largest amplitude vector is then

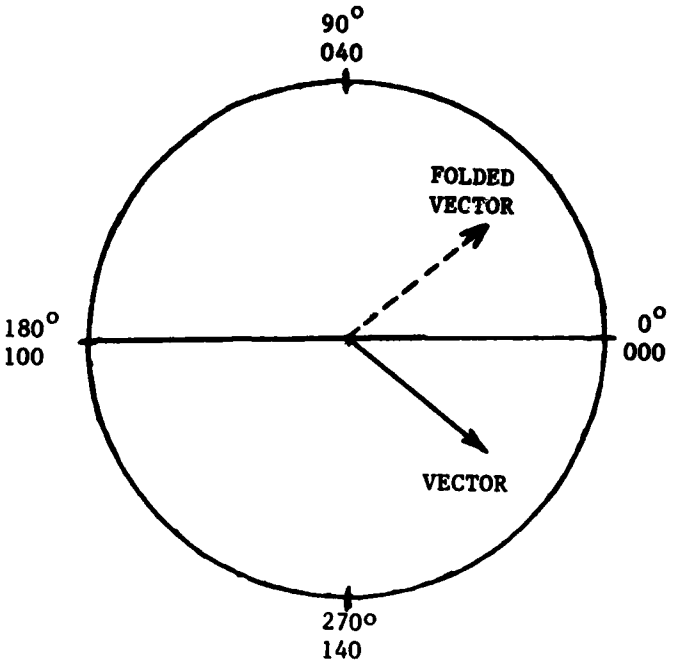
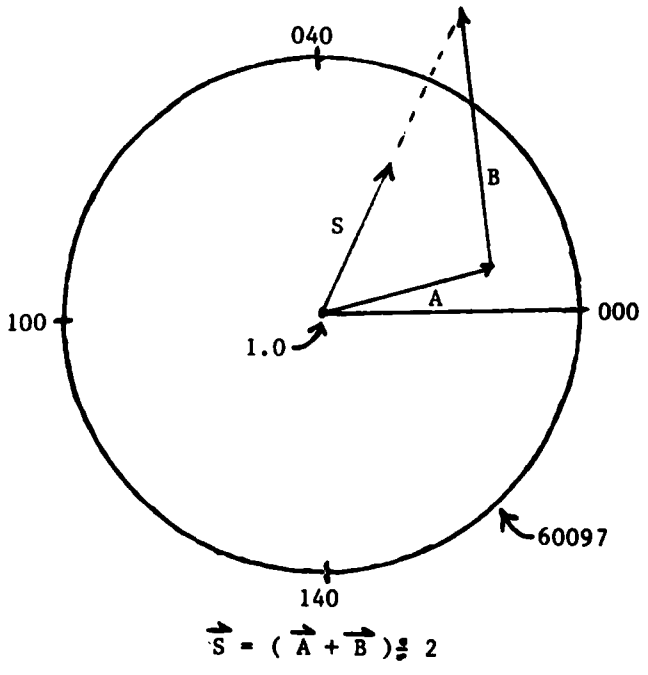


Figure 2. Phase Foldover



$\vec{S} = (\vec{A} + \vec{B}) \div 2$

Figure 3. Normalized Vector Addition

added to perform the inverse normalization process on the phase. This is performed in the "sum" block. No underflow or overflow conditions occur for phase, because the phase quantization is circular. See Figure 2.

The process is now complete; the two vectors have been vertically added and divided by two. The circuit is of reasonable size for a single chip implementation. It would require approximately 800 gates and 24 kilobits of ROM. See Figure 4. This gate complexity is approximately equivalent to an eight-by-eight bit multiplier and an eight bit sine table, but is much more effective. The Vector Adder chip could turn a conventional sixteen bit microprocessor into a low cost complex signal processor capable of performing a FFT butterfly in five operations, without a sine table, with large dynamic range, with minimum storage, and with no overflow concerns. Using an eight-by-eight bit multiplier and sine table, the microprocessor would require more than fifteen operations to perform the FFT butterfly, with limited dynamic range, with maximum storage, and with constant overflow concerns. See Table 2. The FFT butterfly is only used as an example, the vector adder circuit has not been optimized to perform it at the expense of other complex signal processing algorithms.

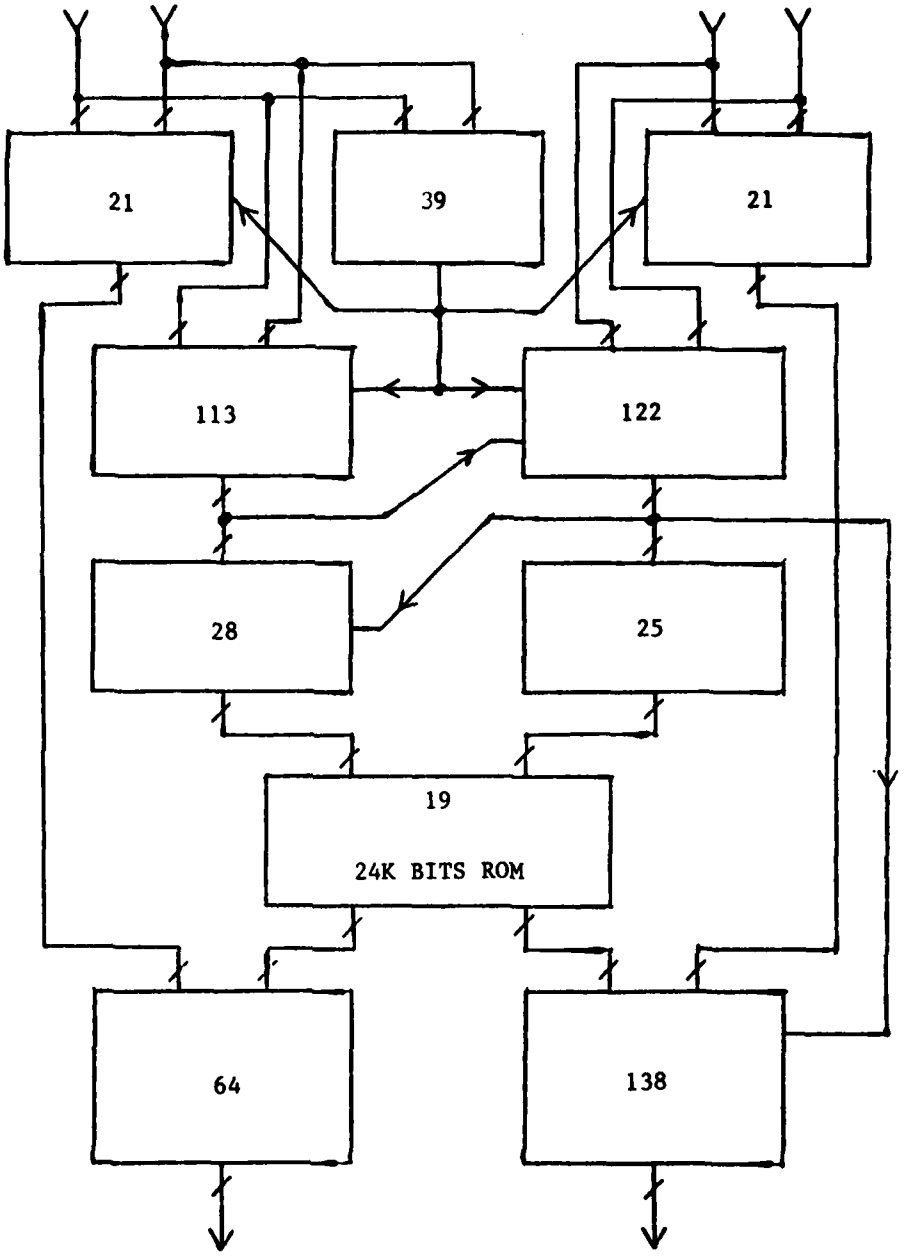


Figure 4. Vector Adder Gate Count

TABLE 2

FFT BUTTERFLY CODING

LOG POLAR ε^{2}	REAL-IMAGINARY *
1. W^N * V1	1. Load WR^N
2. Load V1	2. Load VR1
3. Load V2	3. WR^N * VR1
4. (V1+V2) ÷ 2	4. Load VI1
5. (V1-V2) ÷ 2	5. WR^N * VI1
	6. Load WI^N
	7. WI^N * VI1
	8. Load VR1
	9. WI^N * VR1
	10. WVR1
	11. WVI1
	12. WVR1 + VR2
	13. WVI1 + VI2
	14. WVR1 - VR2
	15. WVI1 - VI2

* Does not include overflow, scaling, or derivation of WR^N and WI^N from sine table.

SECTION III

THE IMPLEMENTATION

There are three basic parts to the implementation of the Log Polar Vector Adder concept. The first is register level simulation. The second is a hardware breadboard and the third is a gate level simulation. The register simulation was done on the "PDP11" computer; the breadboard was connected to the "PDP11," and the gate simulation inputs and outputs were provided to the "PDP11." The "PDP11" system was used to verify the functional equivalence of all three implementations. This was a tremendous aid in the development of all three implementations. The "PDP11" system also performed calculations for the ROM table and initial calculations used during circuit conception. The basic input to all three implementations was a functional layout of the circuit in existing integrated circuits. The functional layout was the controlling medium and functional errors were corrected on this layout.

A. Register Level Simulation

Prior to the register level simulation and during the concept stage, a Fortran program was written and modified to calculate the ROM table. This program is called "Table" and can be found in Appendix A, in its final form. This program calculates the input amplitude quantization as AMPI = 2.0** (-DAI*IA + (1.0/16.0)).

The powers of two made the amplitude logarithmic. "DAI" is the linear delta amplitude for input, and is given as DAI = 1.0/8. "IA" is the input integer amplitude field. This field is the output of the amplitude normalization process and represents ratios less than 1; thus the negative sign in front of "DAI." The "(1.0/16.)" is the one-half least significant bit amplitude offset. The input phase quantization is calculated as PHASI = -DPI*IP-DPI/2. "DPI" is the Delta Phase for input given as DPI 2.0*PI/"200, where "200 is the octal representation of the 128 decimal, and "PI" is the constant 3.1416. The minus sign in front of "DPI" corrects the negative phase normalization as mentioned previously in the functional description. "IP" is the input integer phase field. The field is the output of the phase normalization and phase foldover

operation. The "DPI/2" is the one-half least significant bit of phase offset. This normalized input vector is vectorially added to a vector of one at zero degrees and the results are divided by two. The output amplitude and phase are quantized like the inputs only the one-half least significant bit offsets are not used. However, the output amplitude and phase fields are each offset by minus one to reduce the table size. The output of "Table" was reformatted to form the table for the register level simulation. See "EPROM1" and "EPROM2" in Appendix A. The output was also reformatted for use in the gate level simulation and for programming the EPROM chips of the breadboard.

The register level simulation was derived from the functional layout. FORTRAN Callable "PDP11" assembly language subroutines were written for each functional block. Table 3 lists and describes these subroutines, which can be found in Appendix A. FORTRAN programs simulate the circuit by calling the subroutines in the proper order.

The program "EST" is used to test the register simulation against double precision floating point calculations. Amplitude and phase error limits, "ADLMT" and "PDLMT," are set, and random inputs for the vector adder simulation are generated. Because the vector addition process is logarithmic, the actual errors are a function of input, so the program error limits are exceeded occasionally. An example is the addition of two equal amplitude vectors with a 180 degree phase difference. The errors were manually checked to verify that the vector adder simulation was not at fault.

Three FORTRAN programs were used for the FFT evaluation of the vector adder simulation. These three programs are in Appendix A. The first is called "WEIGHT" and is used to perform a cosine squared weighting of the amplitude. This is used in checkiny the sidelobe performance of the log polar quantization. The second program is called "SHUF." It performs the bit reversed shuffle of data for FFT. The third program is called "STAGE." It performs one stage of the FFT. The complete FFT is performed by making multiple calls of the "STAGE" program. This can be seen in the FFT Batch Job example in Appendix A.

TABLE 3

REGISTER LEVEL SIMULATION SUBROUTINES

Subroutine Name	Function
IDLVA	Determine Largest Vector Amplitude
INA	Normalize Amplitude
INP	Normalize Phase
IRA	Randomize Amplitude
IASL	Amplitude Significance Limit
IPFQ	Phase Foldover Question
IPF	Phase Foldover
INVAT	Normalize Vector ADD Table
ICED	Combine EPROM Data
ISLV	Select Largest Vector
IINA	Inverse-Normalize Amplitude
IVNP	Inverse-Normalize Phase & Inverse Phase Foldover
ICVO	Combine Vector Output
EPROM1	EPROM Table No. 1 (used by INVAT)
EPROM2	EPROM Table No. 2
IBCLR	Clear the 8th Bit (used by STAGE)
IBITR	Bit Reversal Routine
ISWAB	Swap Bytes (used by WEIGHT)

B. Hardware Breadboard

The main objective of the hardware breadboard was to prove the basic circuit design and provide a starting point for the gate level simulation. The breadboard was designed from the functional drawing which gave the integrated circuits and their interconnections. A technician translated this into detailed circuit drawings and a wire wrap board layout drawing. These drawings were then used to fabricate and wire wrap the breadboard. The drawings are given as Figures 5 through 8. Three connections, two for input, and one for output, are for the

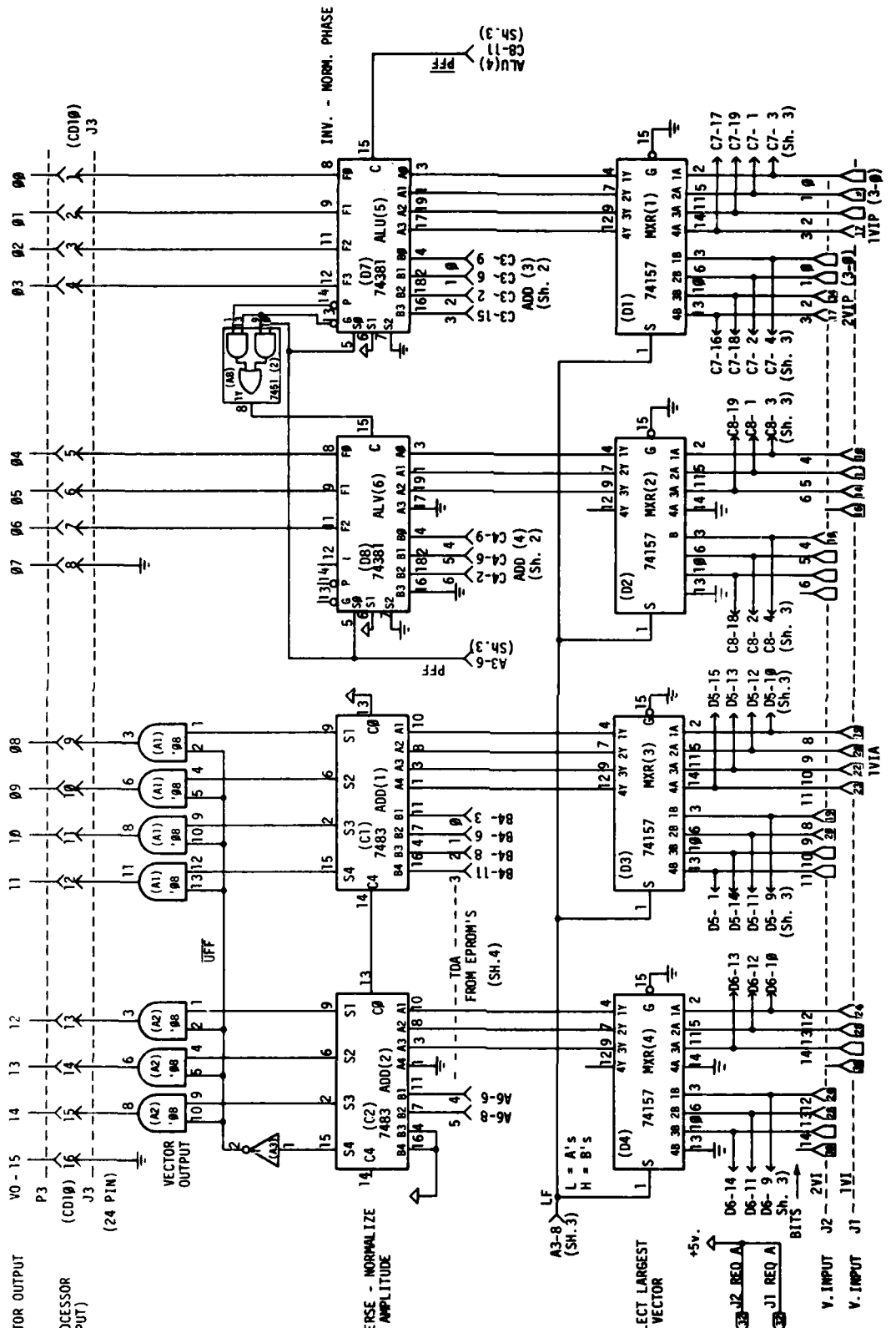


Figure 5. Vector Adder Circuit Drawing Sheet 1

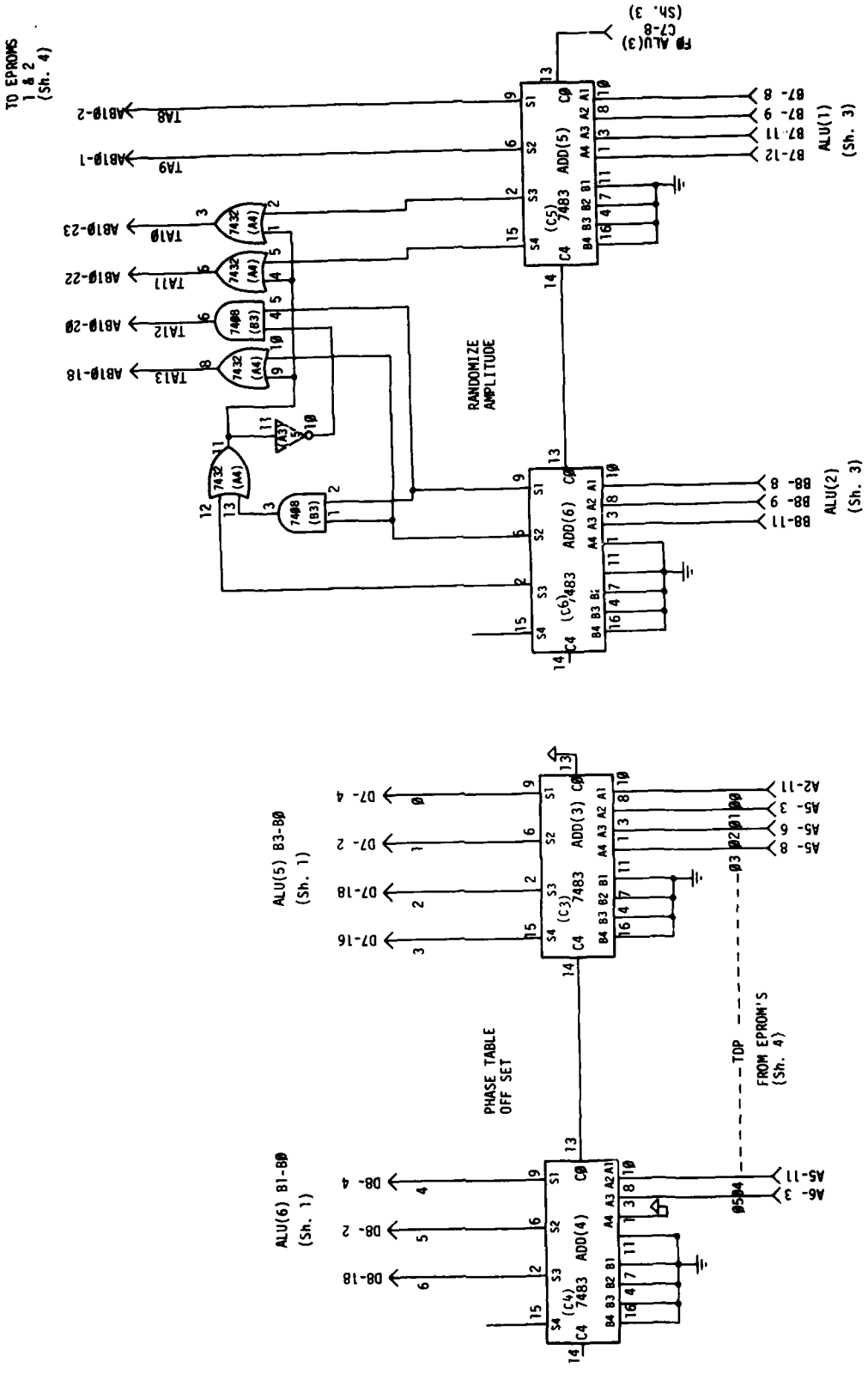


Figure 6. Vector Adder Circuit Drawing Sheet 2

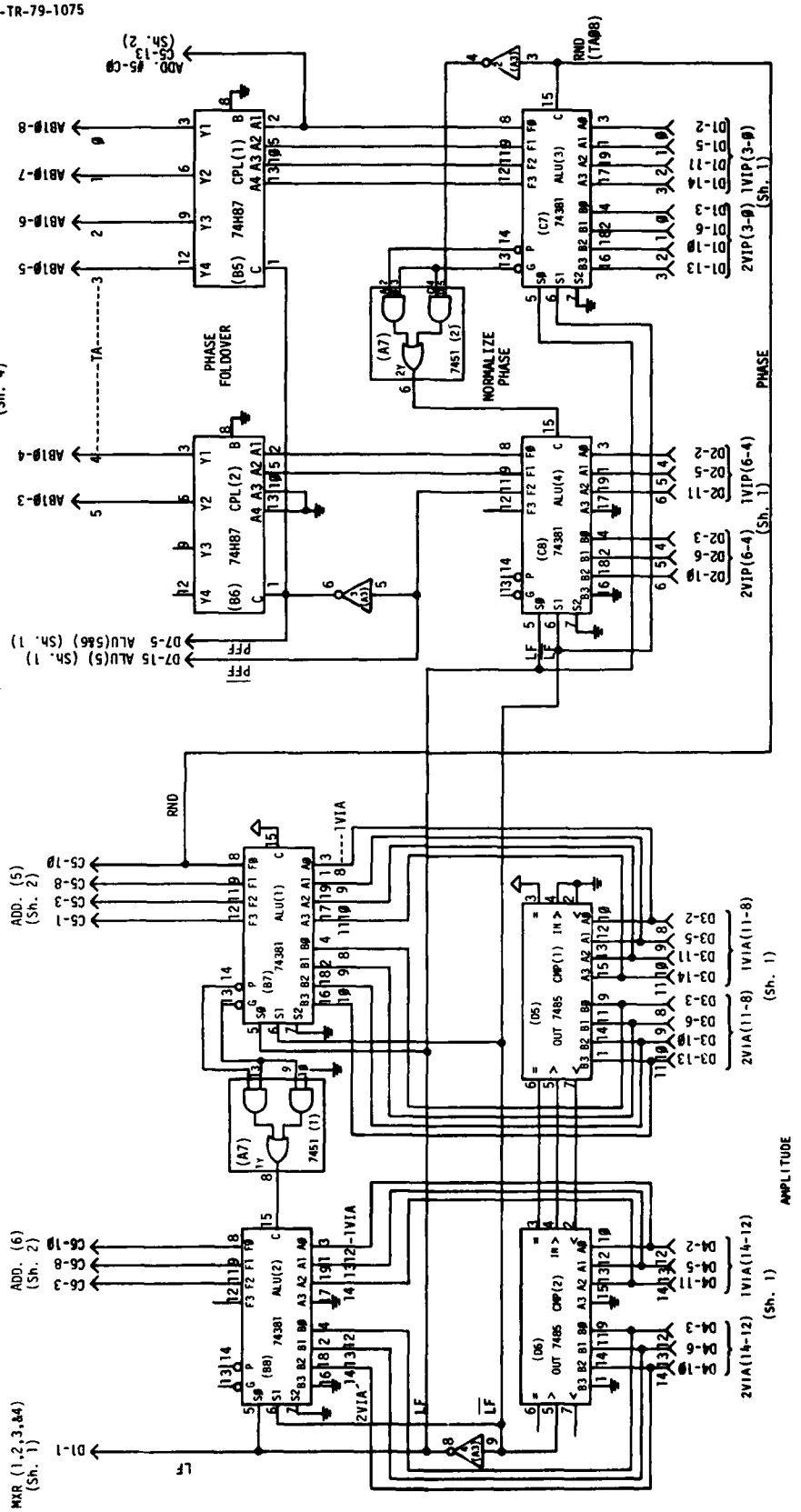


Figure 7. Vector Adder Circuit Drawing Sheet 3

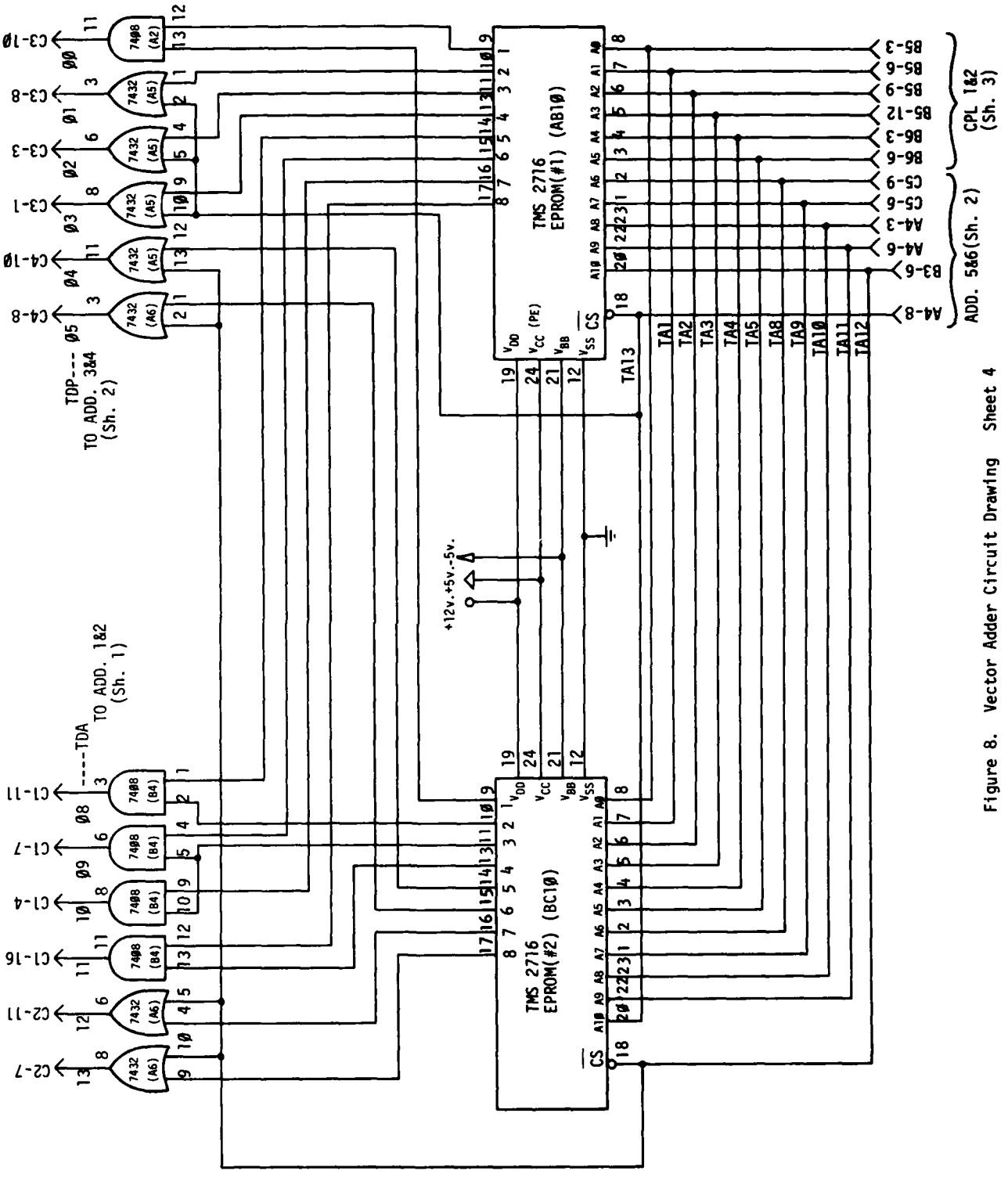


Figure 8. Vector Adder Circuit Drawing Sheet 4

parallel interfaces of the "PDP11." Figure 5 shows the inputs, the outputs, and the multiplexers used for largest vector selection, the amplitude inverse normalization, the amplitude underflow limit, and the phase inverse normalization. Figure 6 shows the phase table offset, which is part of the inverse phase normalization block; the amplitude randomizer, which is part of the amplitude normalize block, and the amplitude significance limit. Figure 7 shows the amplitude comparators, the amplitude normalization, the phase normalization, and the phase foldover. Figure 8 shows the table. This consists of two 2K word-by-8 bit EPROMs and some miscellaneous logic to connect them. Only one 2K-by-8 bit and one 2K-by-4 bit is actually needed for the table. This can be seen examining "EPROM2" in Appendix A. Note that the upper four bits of data never change value.

The hardware only required two hours to debug even though it contained one wrong connection and one bad device. This was mainly due to a professional layout and wire wrap job and the "PDP11's" ability to compare the hardware with the register level simulation and give internal circuit values for fault isolation.

Two FORTRAN programs use the register simulation to test the breadboard through the "DR11-C" parallel interfaces. The first program is called "TESTVA" (Appendix A). The program takes two vectors supplied by the user and calculates their vector addition using the register simulation and the breadboard. The intermediate values of the register simulation are output so that the internal values of the breadboard can be checked. The second program is called "TESTH" (Appendix A). This program compares the breadboard calculations to the register simulation calculations for random inputs. Errors cause the program to stop and wait for operator action. This program was run continuously for four hours and over four million sets of vector additions without an error. An exhaustive test of all the gates in the hardware would have been better, but such a test definition was not worth the effort at this stage in the development.

C. Gate Level Simulation

The main objective of the gate level simulation was the design and checking of a minimum gate count version of the Vector Adder circuit. The resulting gate design is the primary input for any follow-on chip development effort. The starting point for the gate simulation was the breadboard design and its integrated circuit gate layouts. The logic diagrams for each of the TTL circuits used in the breadboard were examined to see which of the gates internal to the IC were needed due to the use or non-use of its input and output lines.

A new block diagram (Figures 9 and 10) was then drawn closely resembling the breadboard design except in those places where there was minimum gate count or a repetitive pattern to the gates.

LOGIC IV (REFERENCE 1), a gate-level simulation/program, was used to test the gate reduced version of the Log-Polar Adder. Each block of Figures 9 and 10 was then made into a LOGIC IV gate level subroutine called a macro (Figures 11-22). In several cases the macro was equivalent to the original TTL circuit.

Two of the more complex circuits were broken down into submacros. These were the 74S381 (Figures 23-29) and the 74LS83 (Figures 30 and 31). Though six 74S381's were used in the breadboard, five unique macros had to be made due to the varied use of the SØ, S1, and S2 control lines (Figures 11-15). Each of the submacros was individually tested before being combined to form the larger macro.

It was impossible to simulate the EPROM exactly, due to the limited ROM capabilities of LOGIC IV. In order to generate the two 2Kx8 bit ROMS thirty-two, 256x4 bit LOGIC IV ROM's were used. This introduced an additional 84 gates to properly decode the ROM's. These gates were not added into the total gate count.

The submacros and macros used in the Logic IV simulation can be found in Appendix A. The following list gives each macro and the description of its function.

A381 - This macro operates exactly as a 74S381 ALU with S2=0, S0=S1 NOT and Cn=0.

B381 - This macro operates exactly as a 74S381 ALU with S2=0, S0=S1 NOT, no A3 or B3 input, and no P, G, and F3 output.

C381 - This macro operates exactly as a 74S381 ALU with S0=S1 and S2=0.

E381 - This macro operates exactly as a 74S381 ALU with the exception of a CN4 output in place of the P and G outputs.

F381 - This macro operates exactly as a 74S381 ALU that uses only 3 sets of its input lines and sets S1=1 and S2=0.

A157 - This macro is the equivalent of a seven bit data selector/ multiplexor (basic chip 74157).

LS83 - This macro is the equivalent of a seven bit adder (basic chip 74S83).

A1X6 - This macro adds one to a six bit value, producing a seven bit result (basic chip 74S83).

A1X7 - This macro adds a one bit input to a seven bit number, producing a seven bit answer (basic chip 74S83).

MH87 - This macro performs a pass or complement function on a six bit value (basic chip 74H87).

MA85 - This macro is exactly equivalent to the 74L85 comparator with (A>B)=0, (A<B) = 0, and (A=B)=1

MB85 - This macro is exactly equivalent to the 74L85 comparator without the most significant two bits.

The following list gives each submacro and the description of its function.

SB32, SB33, SB36 - These submacros produce the equivalent F0 output of the 74S381 ALU when S2=0 and S0=S1 NOT.

SB42, SB43, SB45, SB46 - These submacros produce the equivalent F1 output of the 74S381 ALU when S2=0 and S0=S1 NOT.

SB52, SB53, SB55, SB56 - These submacros produce the equivalent F2 output of the 74S381 ALU when S2=0 and S0=S1 NOT.

SB63 - This submacro produces the equivalent F3 output of the 74S381 ALU when S2=0 and S0=S1 NOT.

SB15, SB16 - These submacros are equivalent to one set of input lines on the 74S381. They give the intermediate results when S1=1, S2=0, and Cn=S0 NOT.

SB35 - This submacro uses the intermediate results from SB15 to generate the F0 equivalent output of a 74S381 ALU when S1=1, S2=0, and Cn=S0 NOT.

SB65 - This submacro uses the intermediate results from SB15 to produce the F3 equivalent output of a 74S381 ALU when S1=1, S2=0, and Cn=S0 NOT.

SB11, SB12, SB13 - These submacros are equivalent to one set of input lines on the 74S381 ALU without the internal inverted inputs. They produce an intermediate result when S2=0, S0=S1 NOT, and Cn=0.

SB41 - This submacro uses the intermediate results from SB11 to produce the F1 equivalent output of the 74S381 ALU when S2=0, S0=S1 NOT, and Cn=0.

SB51 - This submacro uses the intermediate results from SB11 to produce the F2 equivalent output of the 74S381 ALU when S2=0, S0=S1, and Cn=0.

SB61 - This submacro uses the intermediate results from SB11 to produce the F3 equivalent output of the 74S381 ALU when S2=0, S0=S1 NOT, and Cn=0.

SA83 - This submacro is the equivalent of the input gates of a 74LS83 Adder.

SB83 - This submacro is the equivalent of the input gates of a 74LS83 Adder when C0=1.

SC83 - This submacro is the equivalent of the input gates of a 74LS83 Adder when the B input is equal to one.

When all of the macros were written and debugged the main program (see Appendix A) was written by combining all of the macros. Due to its size and complexity, nearly all of the main computer's memory space and over 22 minutes of CPU time were needed to simulate 512 Vector additions.

The register level simulation on the "PDP11" was then rerun using the identical inputs as those used in the LOGIC IV simulation. Both sets of results were then stored on a disk file. They were then edited so their formats were identical, after which a "source compare" was run, and no errors were detected.

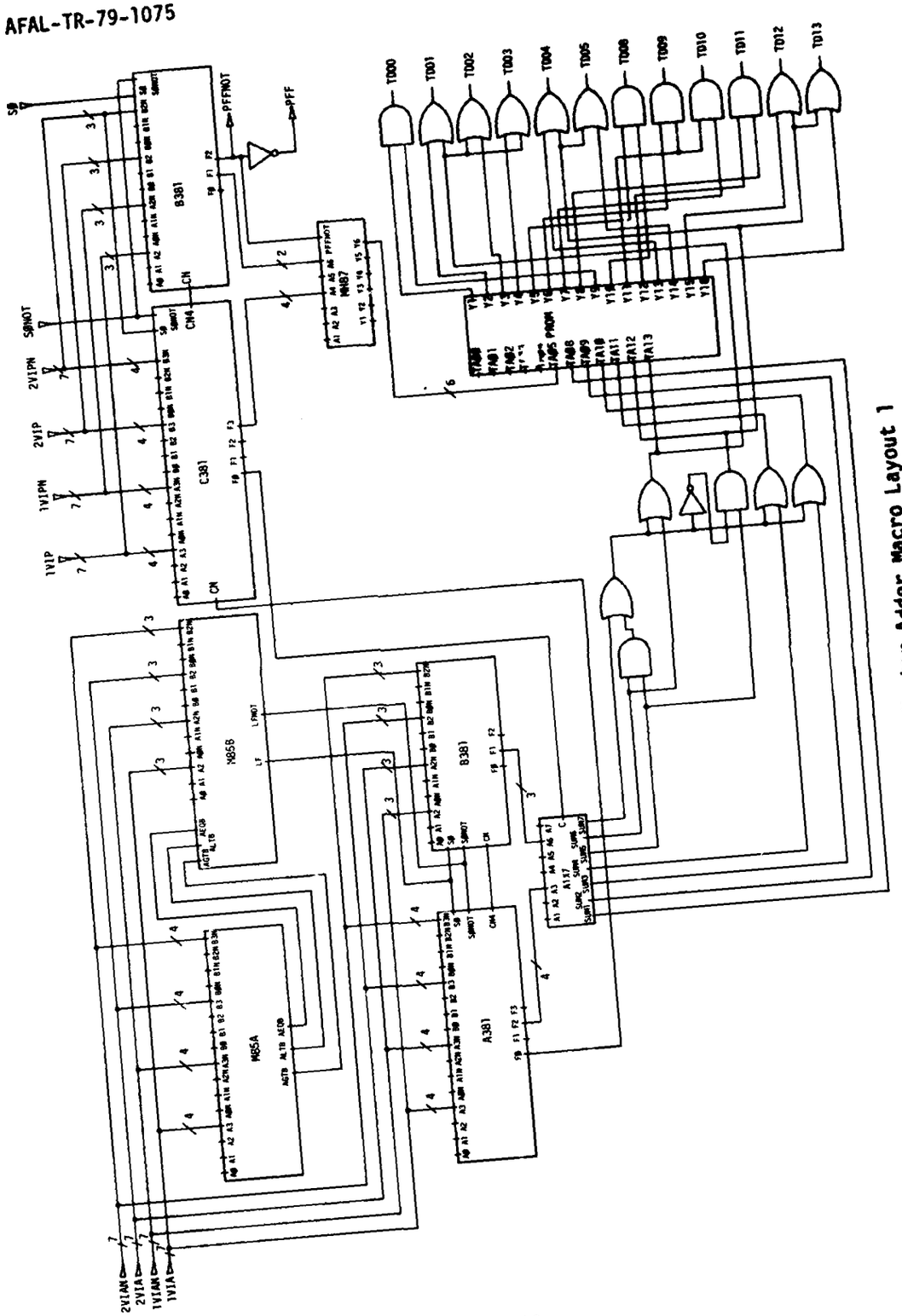


Figure 9. Vector Adder Macro Layout 1

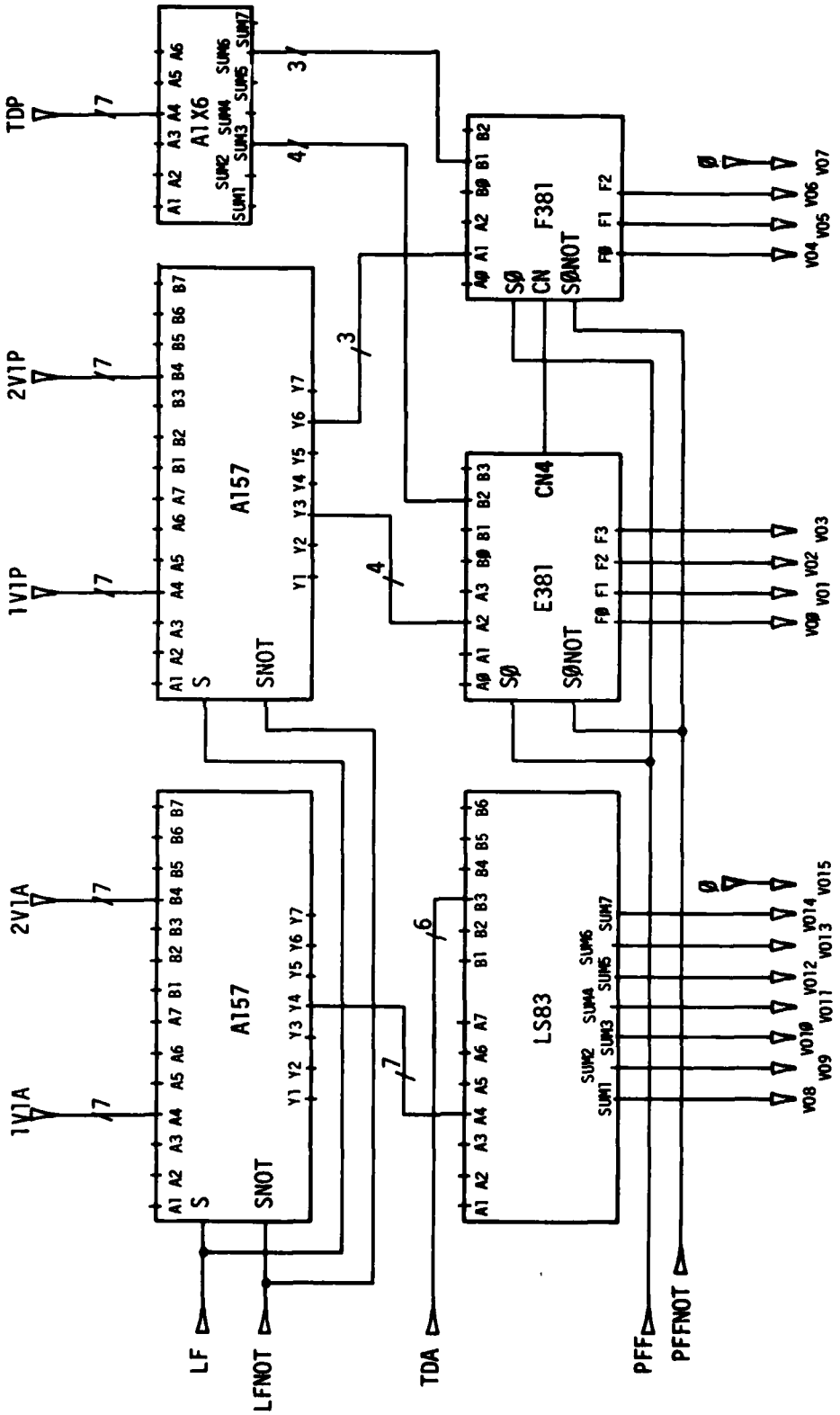


Figure 10. Vector Adder Macro Layout 2

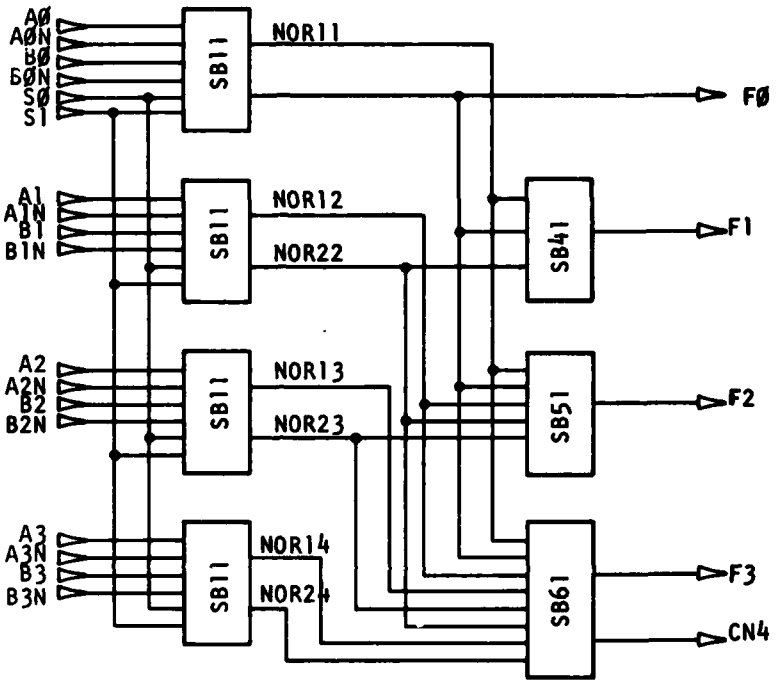


Figure 11. A381 Macro

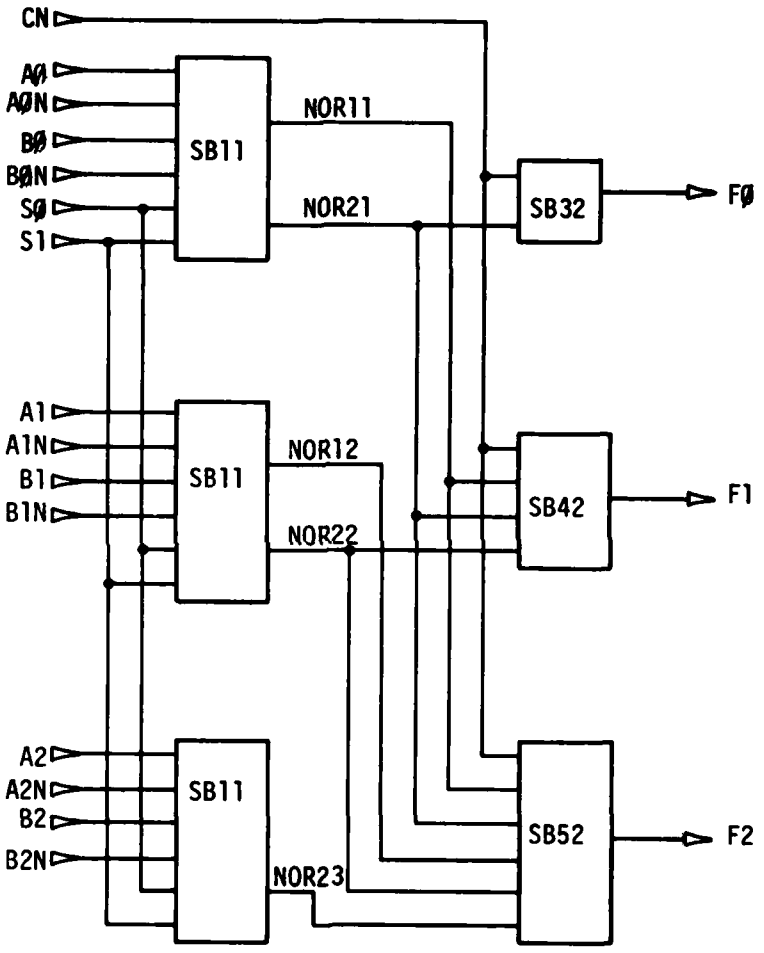


Figure 12. B381 Macro

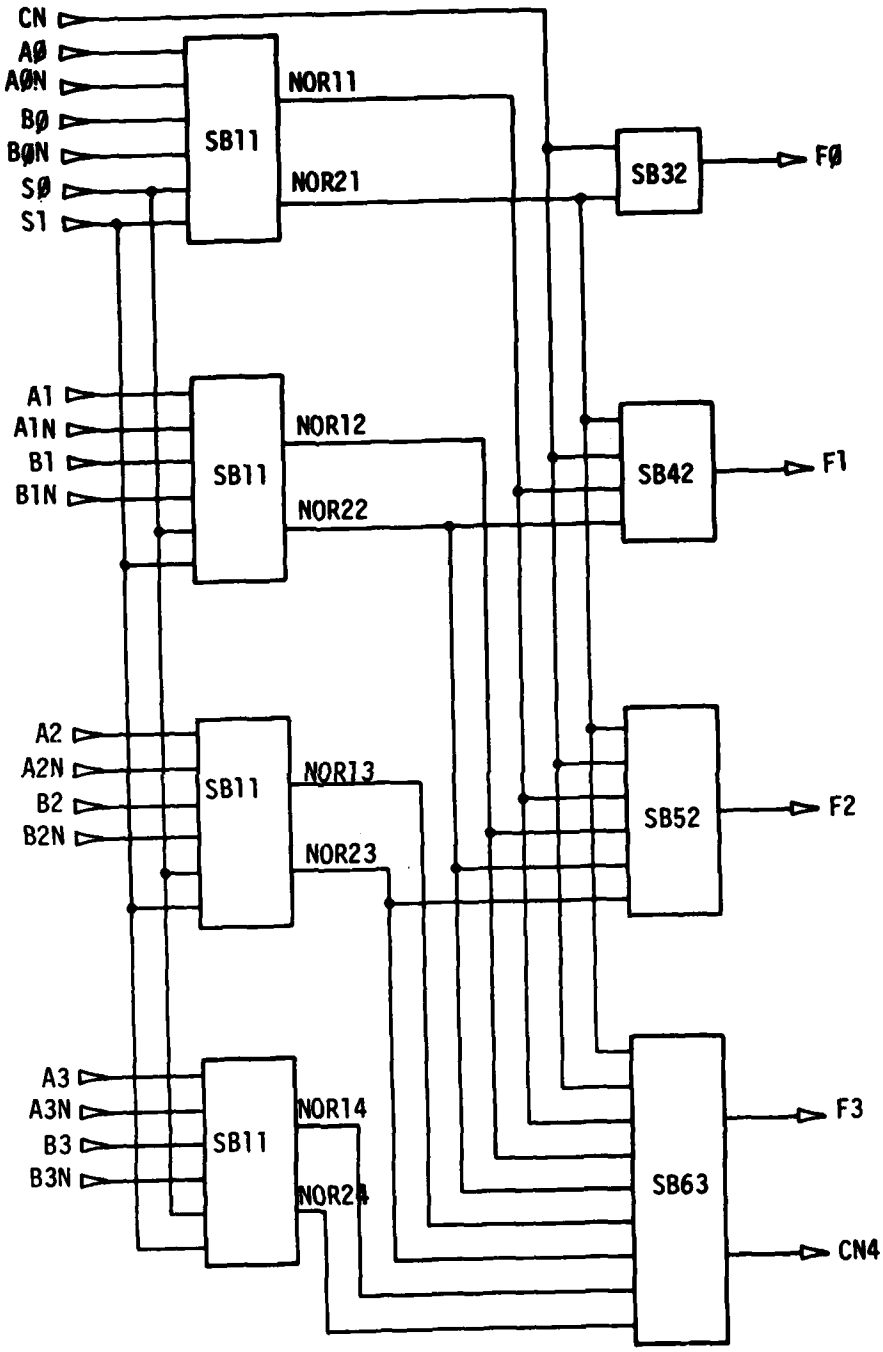


Figure 13. C381 Macro

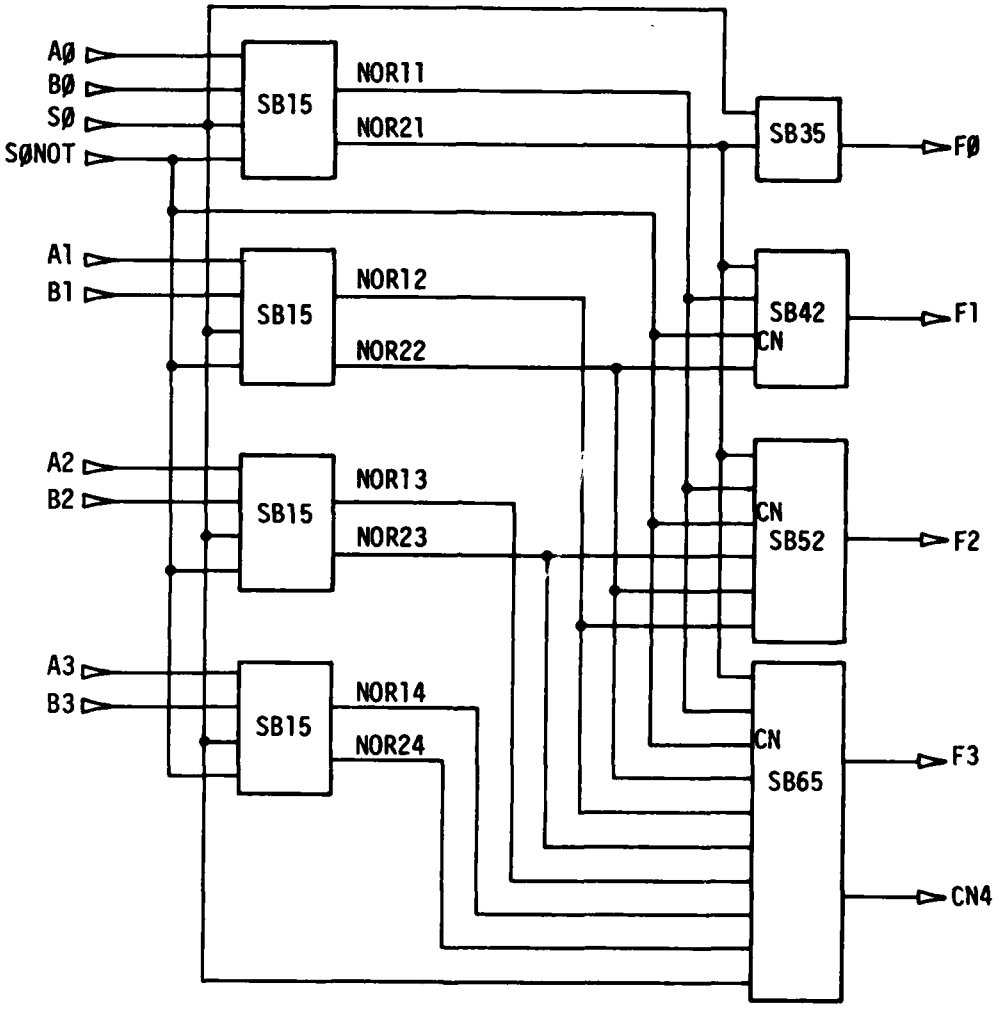


Figure 14. E381 Macro

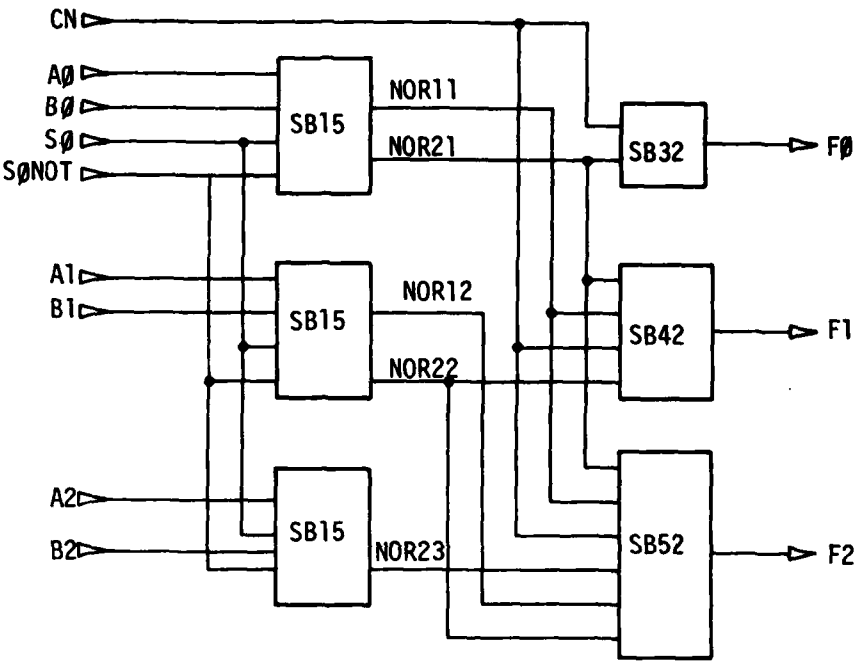


Figure 15. F381 Macro

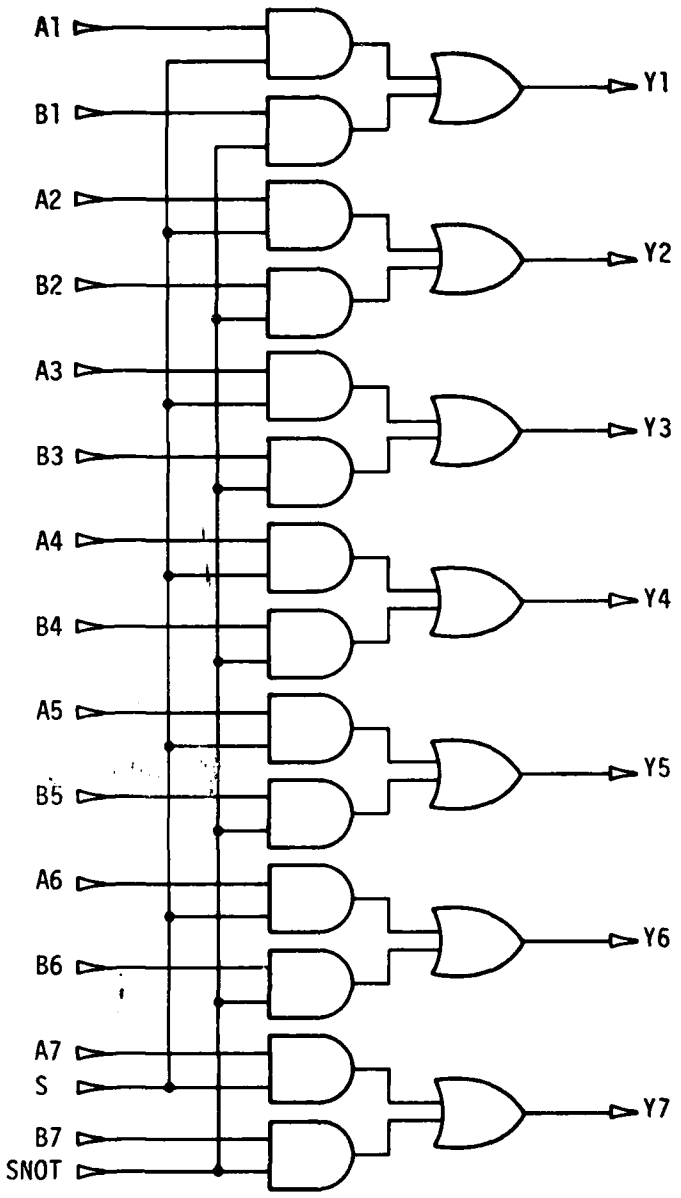


Figure 16. A157 Macro

LS83

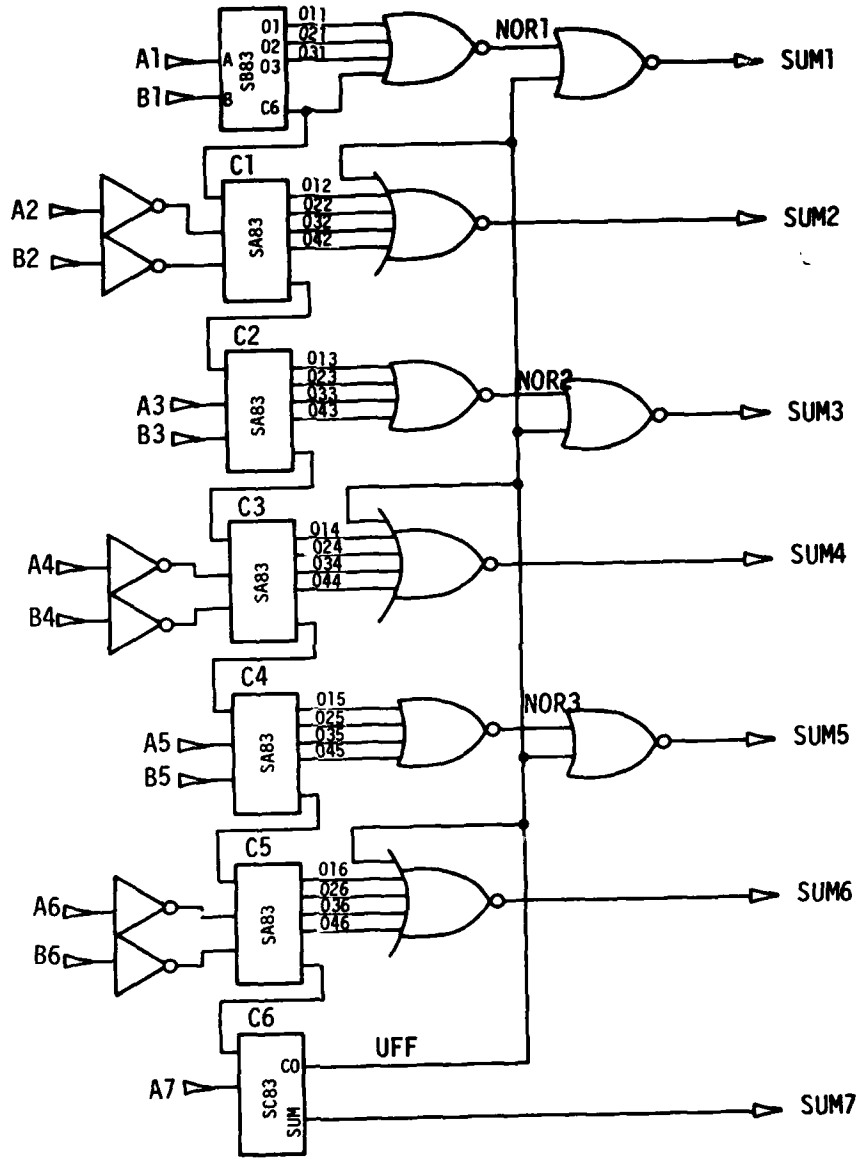


Figure 17. LS83 Macro

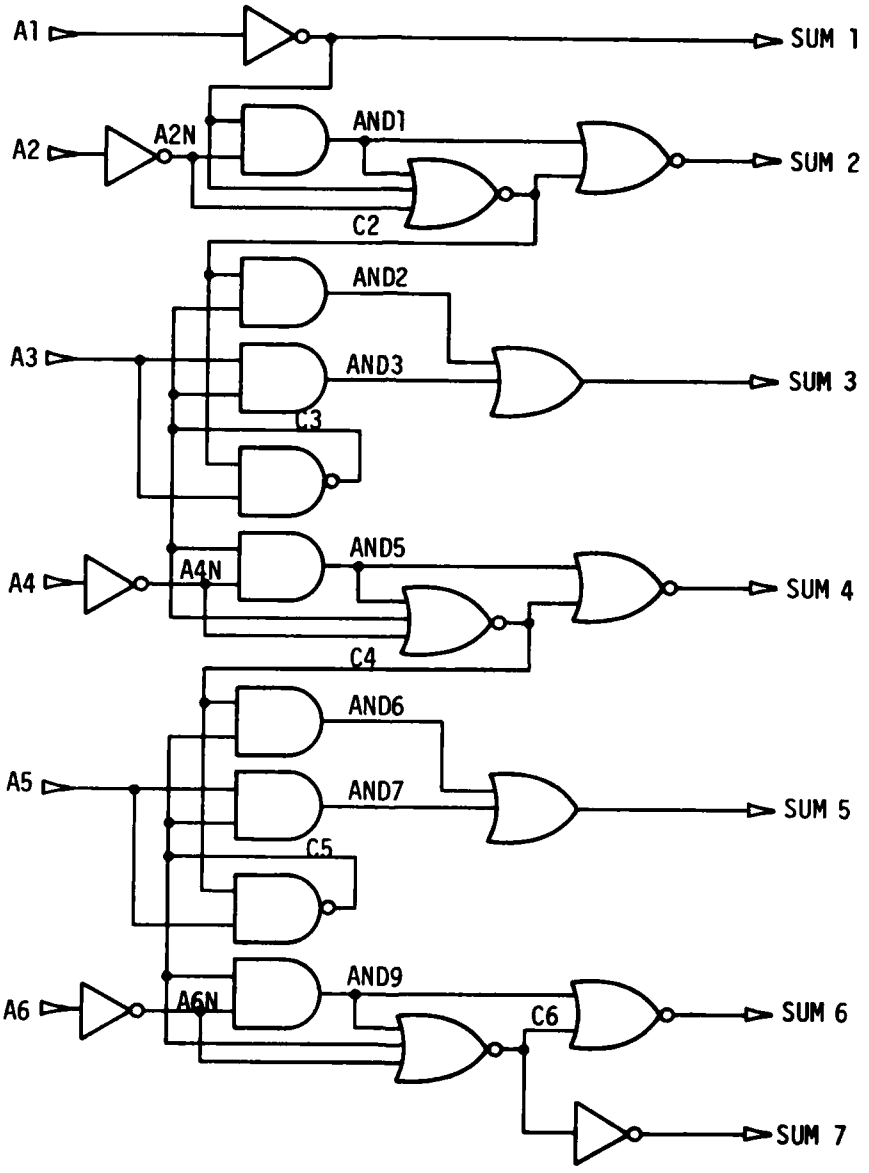


Figure 18. A1X6 Macro

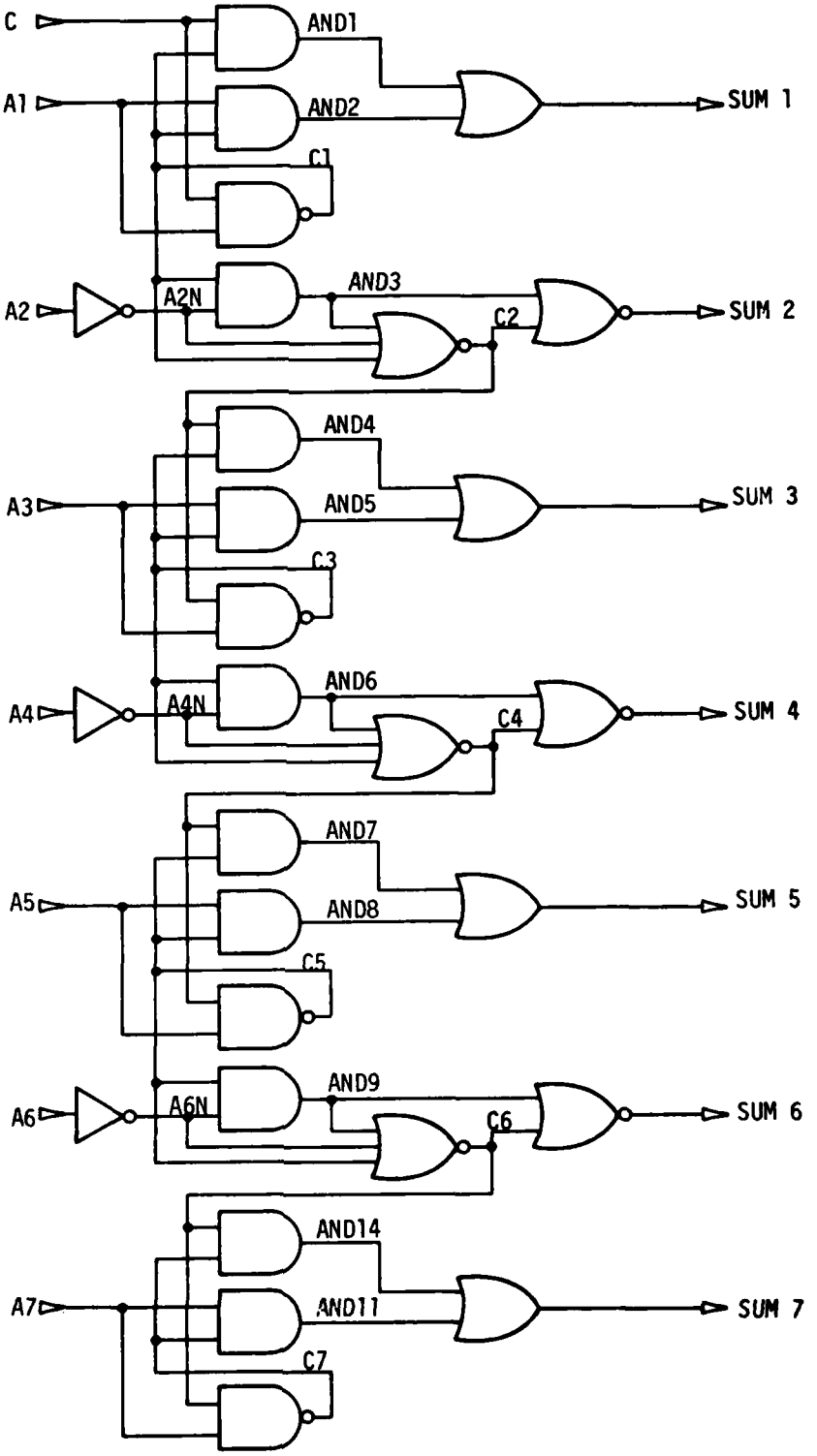


Figure 19. A1X7 Macro

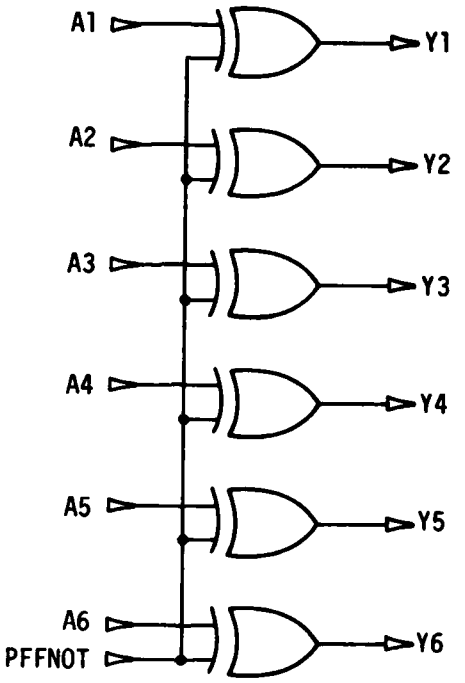


Figure 20. MH87 Macro

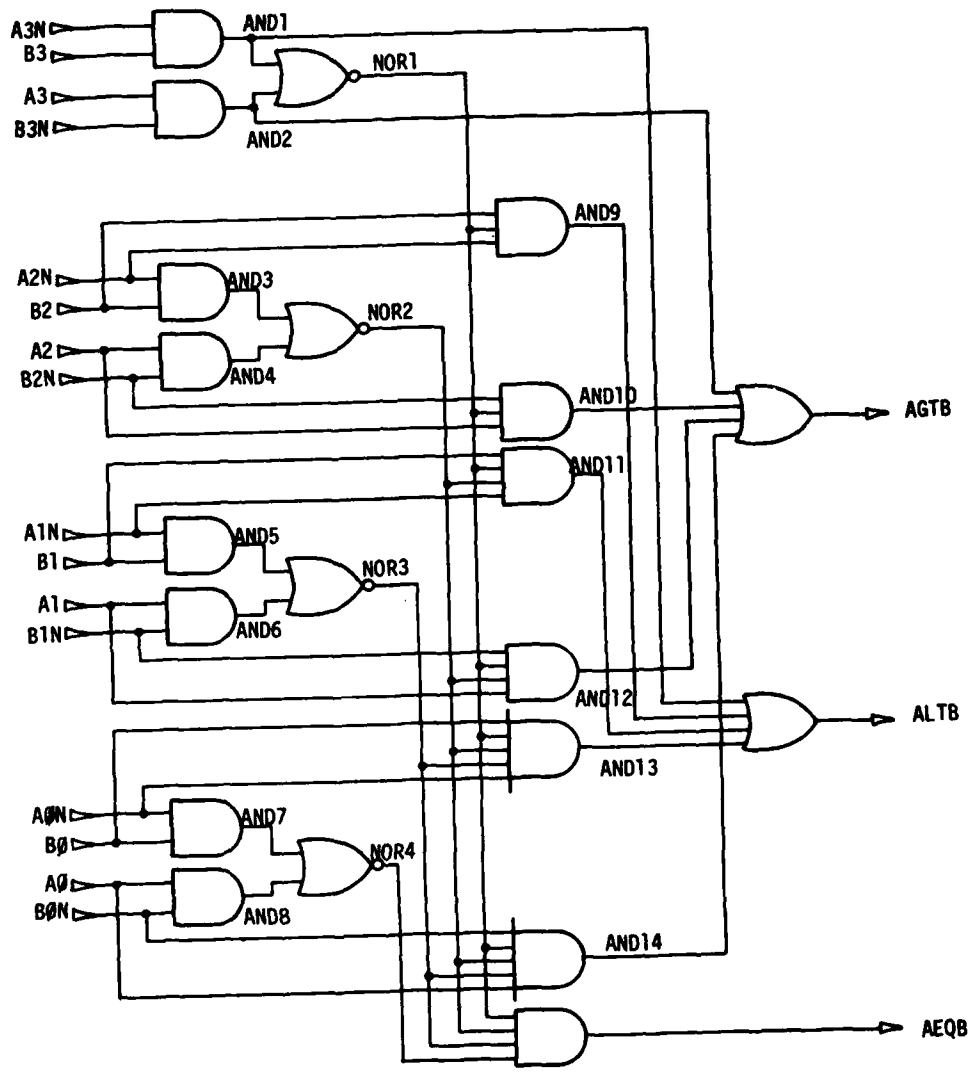


Figure 21. MA85 Macro

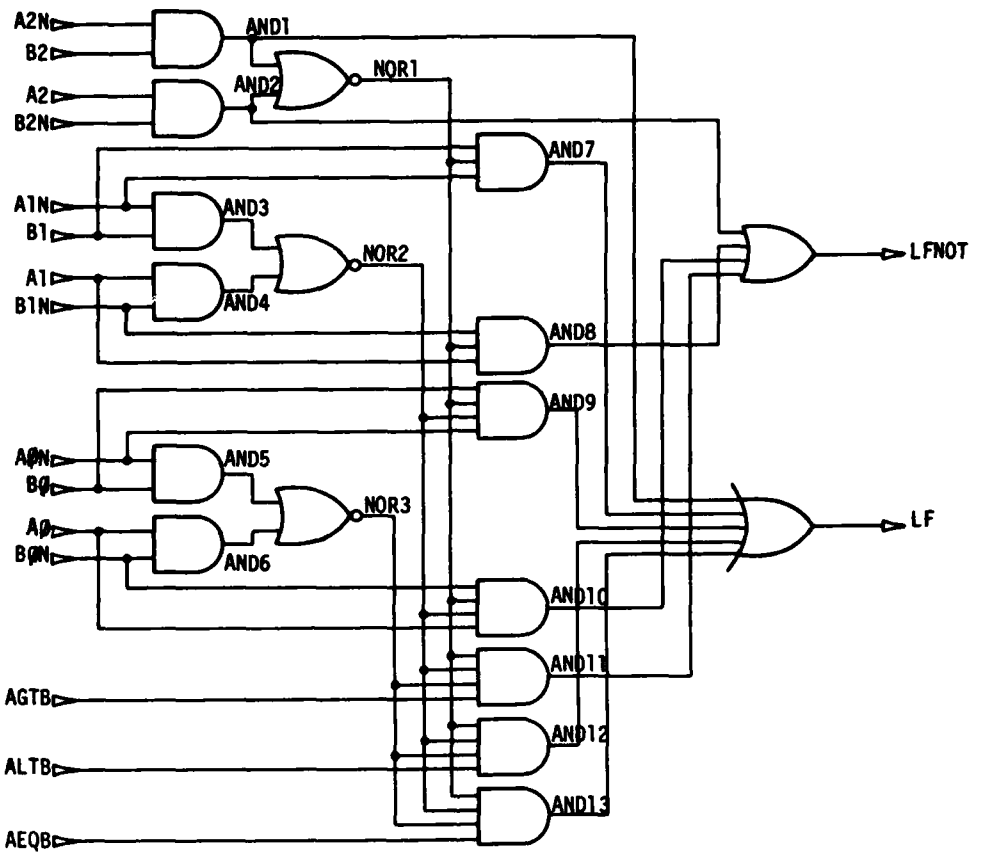


Figure 22. MB85 Macro

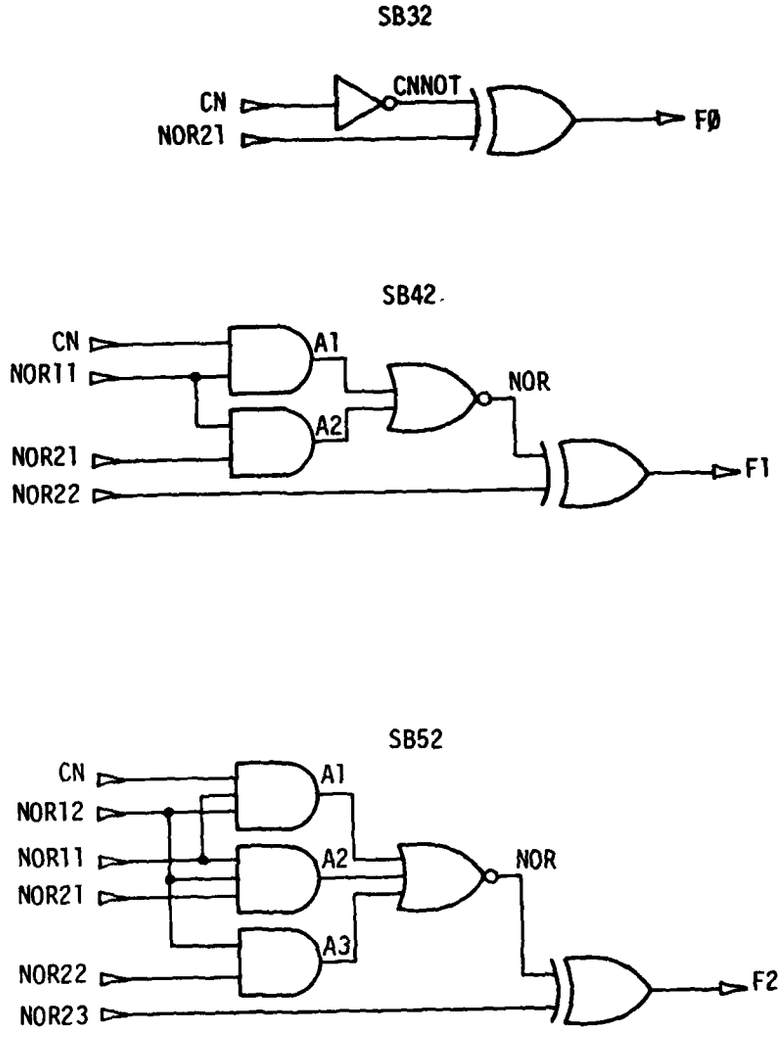


Figure 23. SB32, SB42, and SB52 Submacros

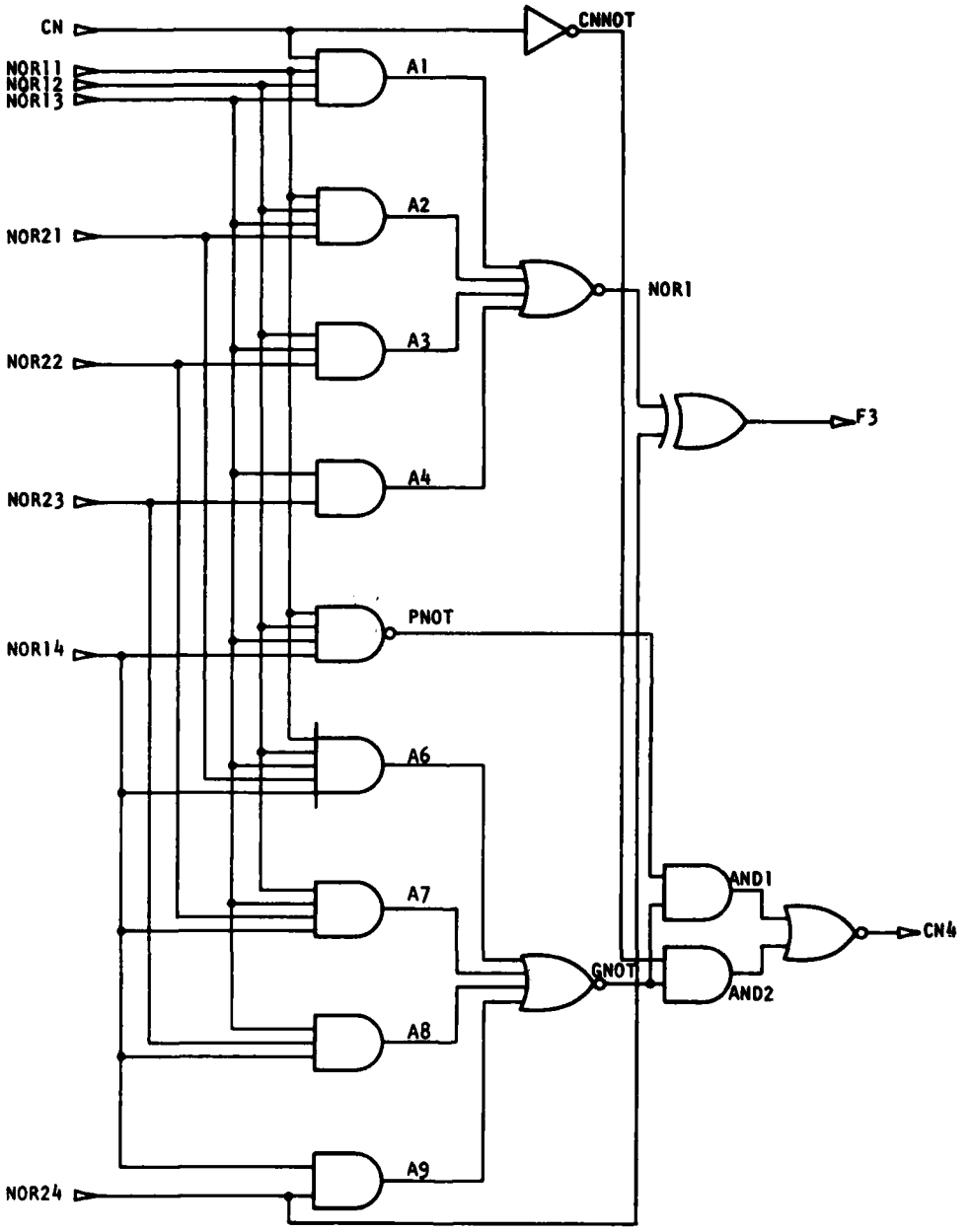


Figure 24. SB63 Submacro

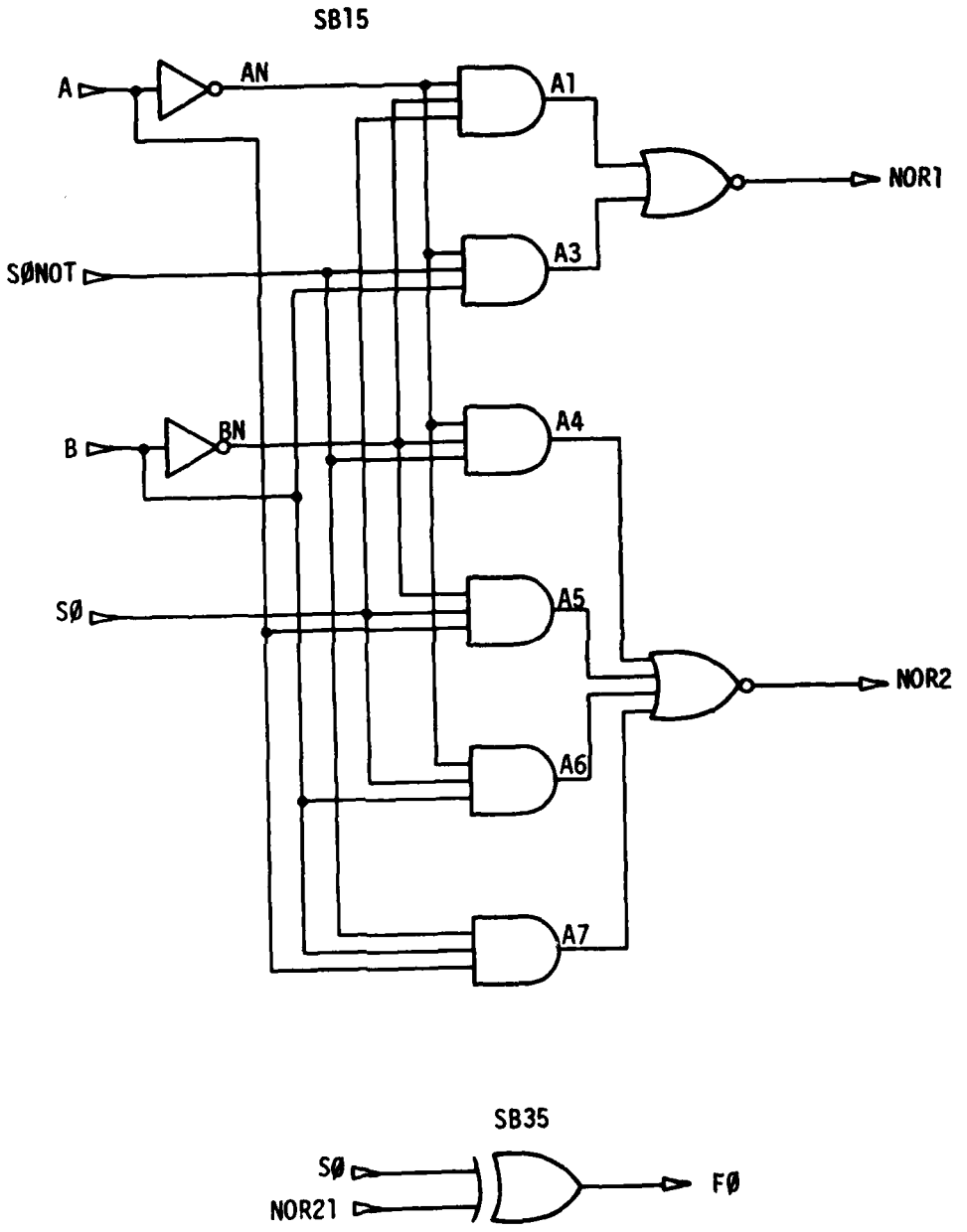


Figure 25. SB15 and SB35 Submacros

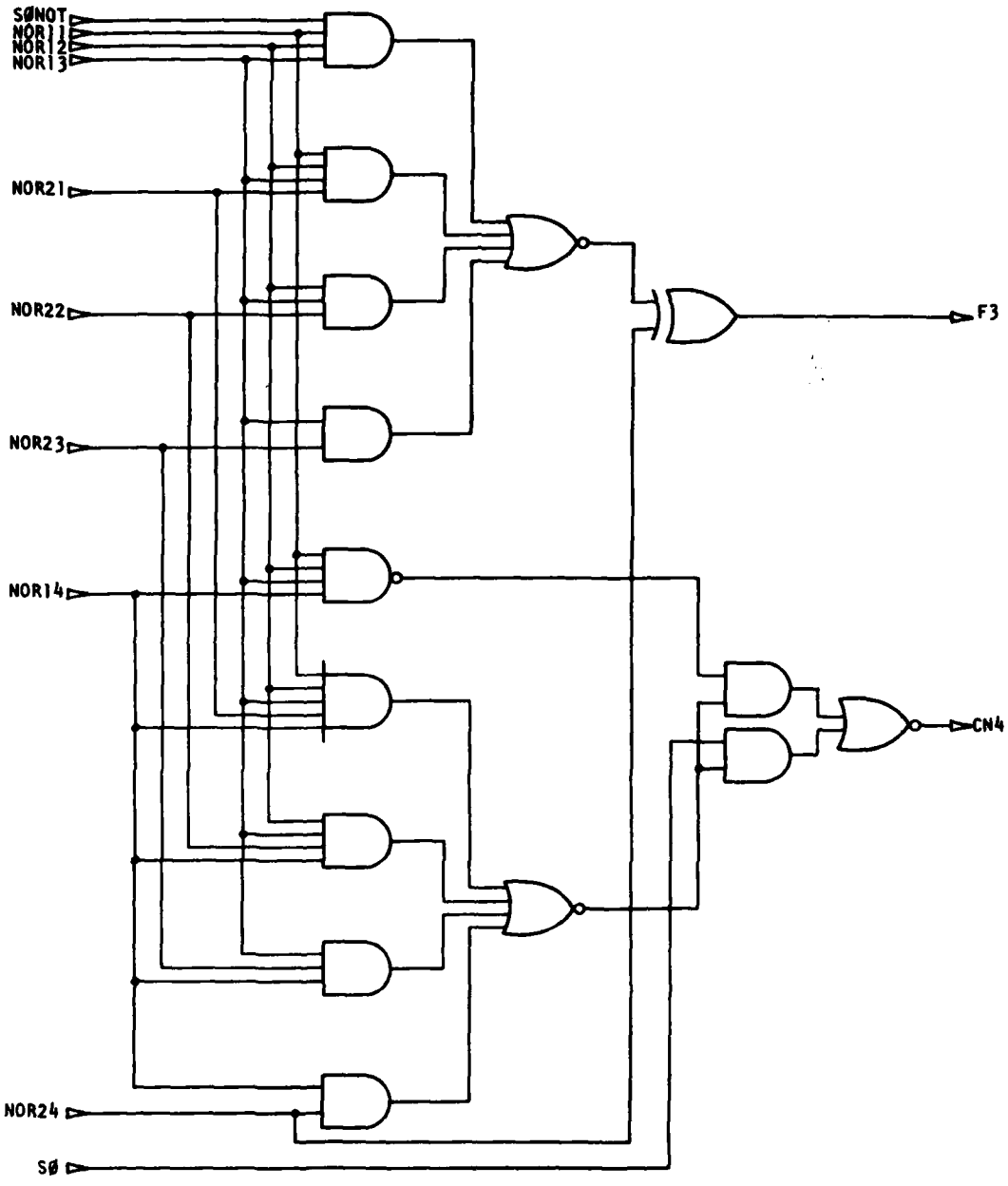


Figure 26. SB65 Submacro

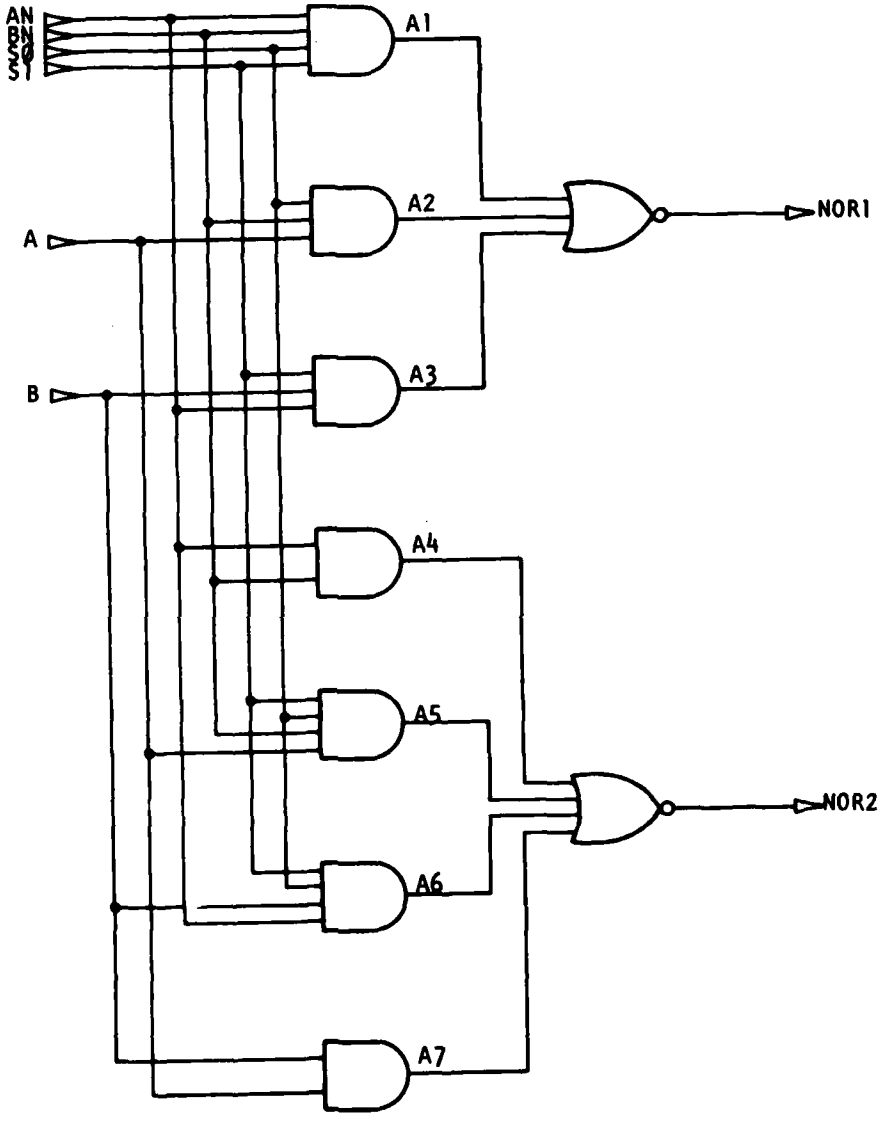


Figure 27. SB11 Submacro

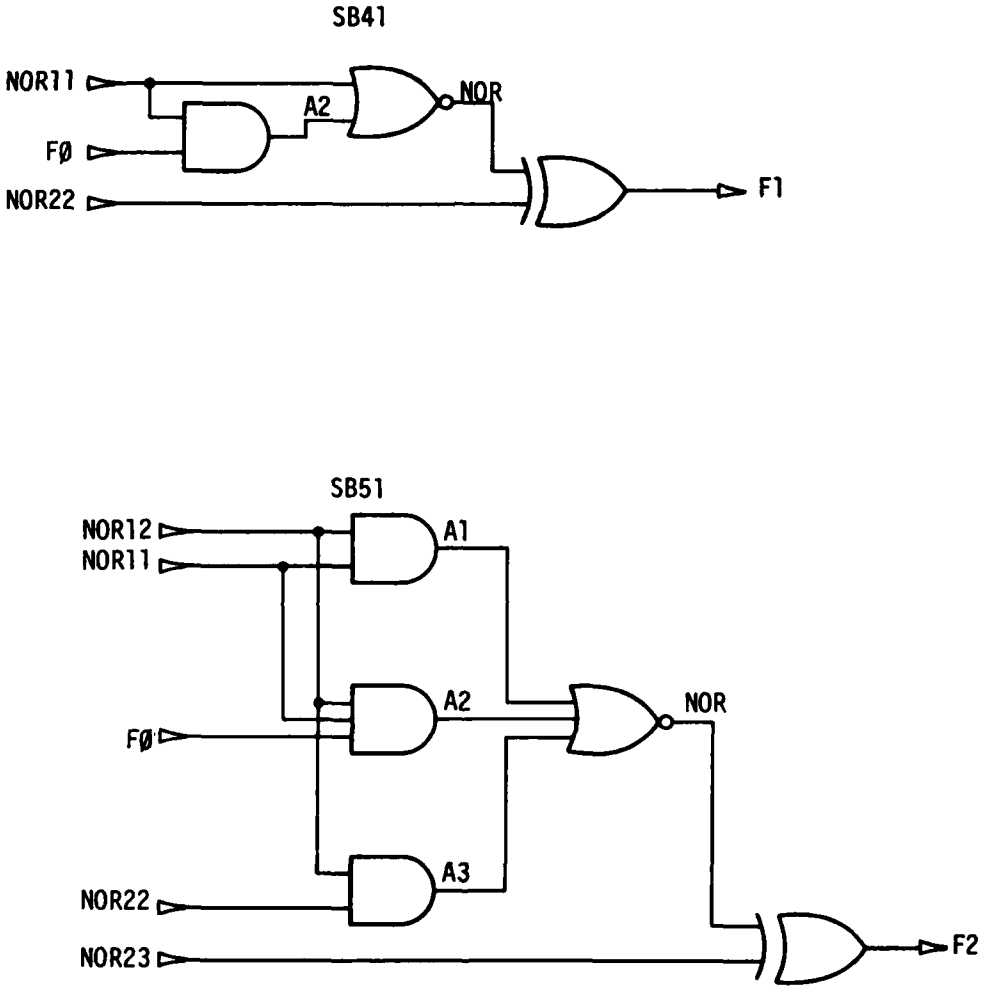


Figure 28. SB41 and SB52 Submacros

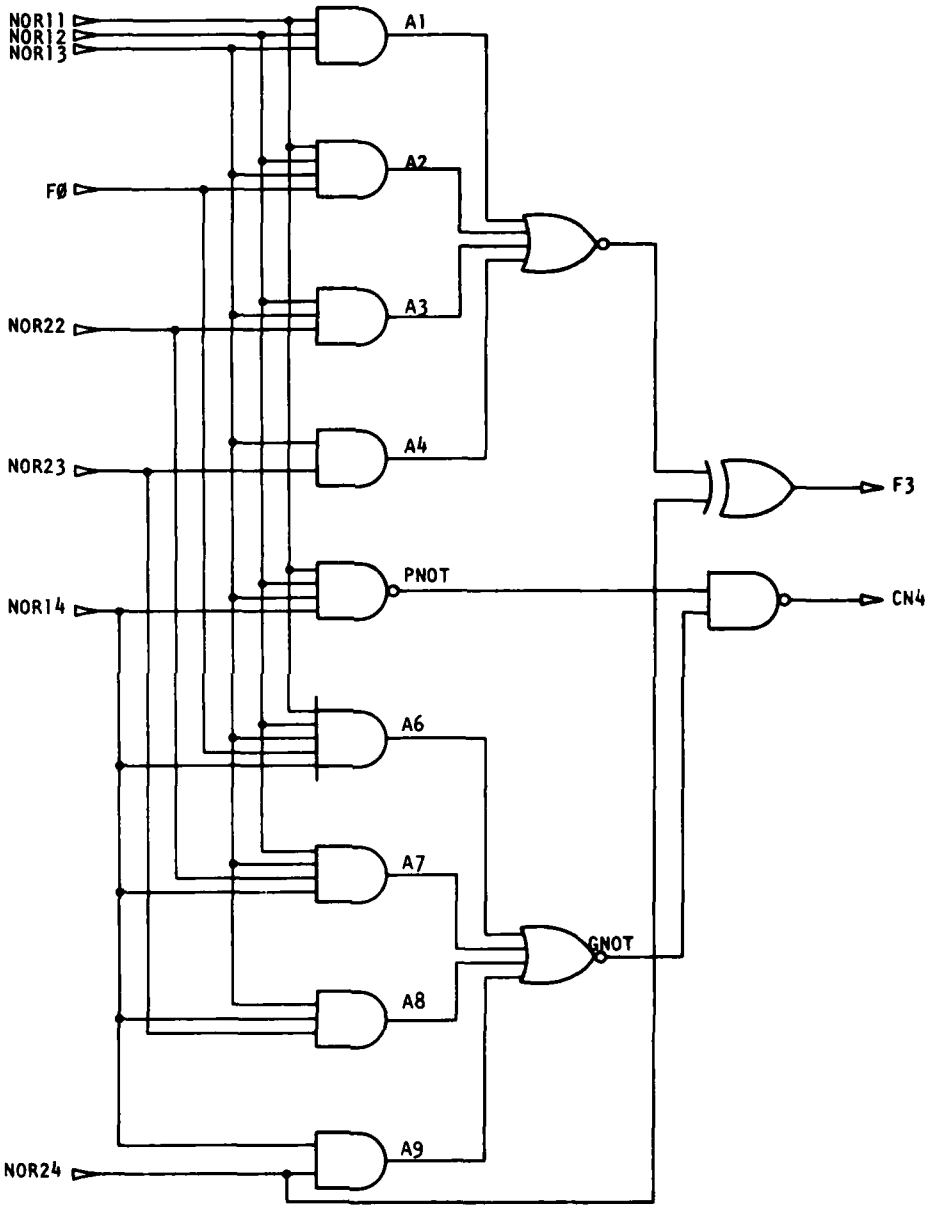


Figure 29. SB61 Submacro

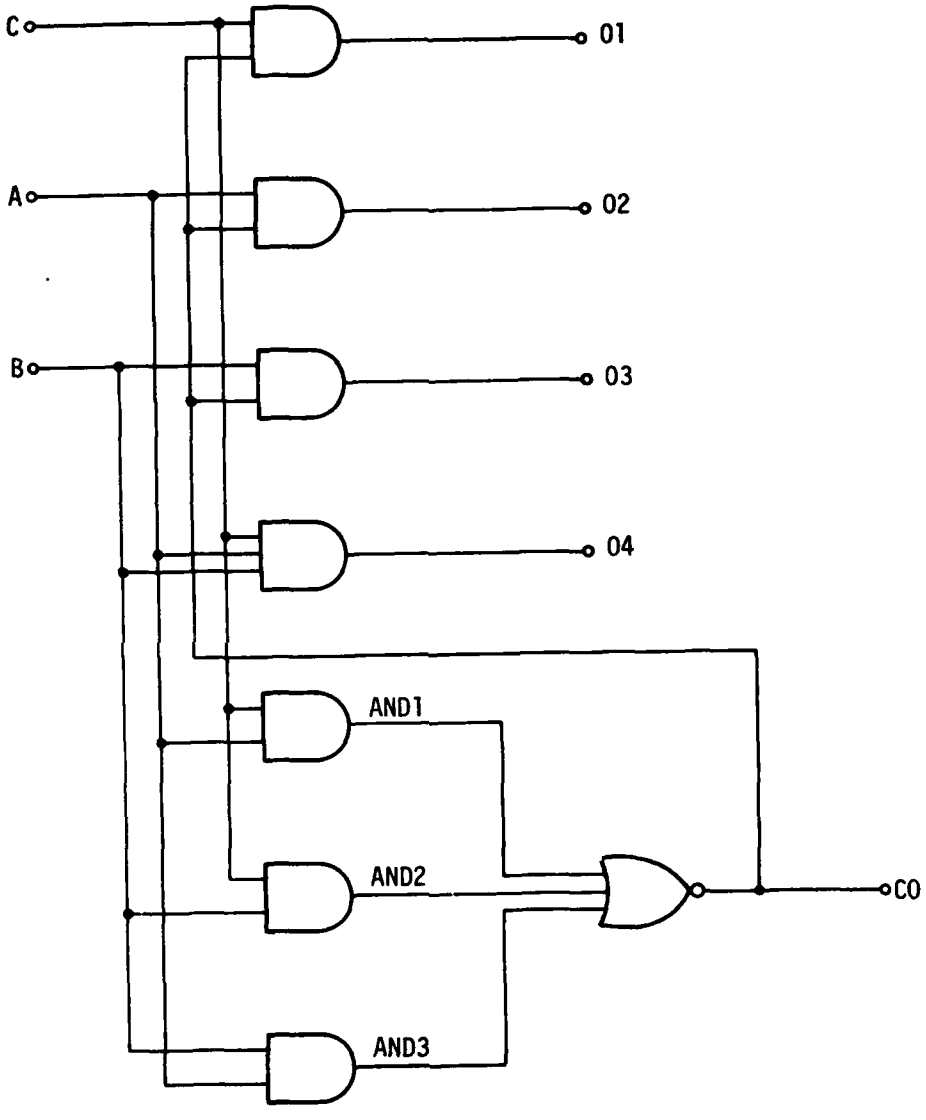
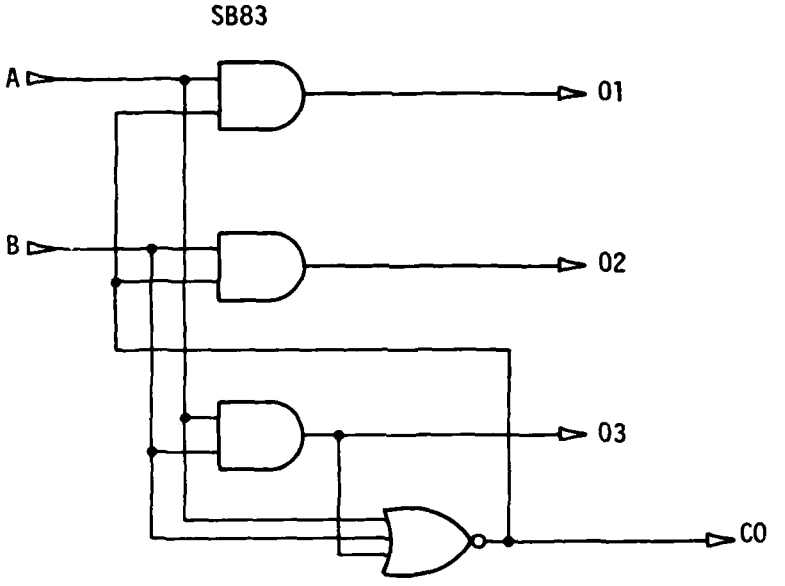


Figure 30. SA83 Submacro



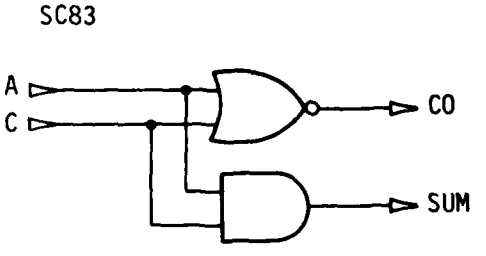


Figure 31. SB83 and SC83 Submacros

SECTION IV

CONCLUSIONS

Unfortunately, the Log Polar Quantization scheme is presently a solution looking for the appropriate problem. Such a complex signal processing problem would require low cost, and only a limited processing accuracy, i.e., spurious signals greater than 30 db down and quantization noise greater than 38 db down. Although Log Polar Quantization offers significant savings in digital hardware, its main impact will be in simplifying the IF to digital, and digital to IF conversion process. The digital hardware savings will primarily come from the exploitation of commercial microprocessors. The simplified IF-to-digital conversion will come from the development of a phase-to-digital converter which employs a digital phase-lock loop for conversion. The simplified digital-to-IF conversion will come from the development of suitable monolithic digital phase modulators that operate on a digital phase input.

APPENDIX A

LOG POLAR QUANTIZATION

COMPUTER PROGRAMS

TABLE OF CONTENTS

```
      PROGRAM TABLE
      DIMENSION IDATA(4096)
      REAL*8 PI,ALOG2,DPI,DPO,DAI,DAO,AMPI,PHASI
      REAL*8 XR,XI,AMPO,PHASO,TEMP
      PI=3.14159265359
      ALOG2=DLOG(2.0)
      DPI=2.0*PI/'200
      DPO=2.0*PI/'200
      DAI=1.0/8
      DAO=1.0/8
      PRINT 102,PI,DPI,DPO,DAI,DAO
      IA=-1
      II=0
      DO 100 IAL=1,'55
      IA=IA+1
      AMPI=-DAI*IA+(1.0/16.0)
      AMPI=2.0**AMPI
      IP=-1
      DO 110 IPL=1,'100
      II=II+1
      IP=IP+1
C*************************
      IBI=IPACK(IP,IA)
C*************************
      PHASI=-DPI*IP-DPI/2
      XR=1.0+AMPI*DCOS(PHASI)
      XI=AMPI*DSIN(PHASI)
      AMPO=DSQRT(XR**2+XI**2)/2
      PHASO=0.0
      IF(XR.NE.0.0) PHASO=DATAN2(XI,XR)
      TEMP=PHASO+DPO/2
      IF(TEMP.LT.0.0) TEMP=TEMP+2.0*PI
      IPO=IDINT(TEMP/DPO)
      IF(IPO.EQ.'200) IPO='0
      IF(IPO.GT.'177.OR.IPO.LT.0) PRINT 103,IPO
      IAO=-'177
      IF(AMPO.EQ.0.0) GO TO 115
      TEMP=DLOG(AMPO)/ALOG2+DAO/2.0
      IF(TEMP.LT.0.0) TEMP=TEMP-DAO
      IAO=IDINT(TEMP/DAO)
      IF(IAO.GT.'177) PRINT 104,IAO
      IF(IAO.LT.-'177) IAO=-'177
115   CONTINUE
C*****************************
C
      IF(IPO.NE.0) IPO=IPO-1
      IF(IPO.EQ.0) IPO='177
      IF(IAO.NE.0) IAO=IAO-1
```

```
      IF(IAO.EQ.0) IAO="377
C
      IBO=IPACK(IPO,IAO)
C*****************************
      WRITE(4,101) IBI,AMPI,PHASI,AMPO,PHASO,IBO
      IDATA(II)=IBO
110   CONTINUE
100   CONTINUE
      IISTA=-"7
      IISTO="0
      DO 130 IAL=1,"54
      DO 120 IPL=1,"10
      IISTA=IISTA+"10
      IISTO=IISTO+"10
      WRITE(2,105) (IDATA(II),II=IISTA,IISTO)
120   CONTINUE
      WRITE(2,106)
130   CONTINUE
105   FORMAT(8X,5H.WORD,1H ,O6,7(1H,,O6))
106   FORMAT(1X)
      STOP
101   FORMAT(1X,O6,2(1X,F8.4,1X,F8.4),1X,O6)
102   FORMAT(1X,6HPI   = ,E12.5,/1X,6HDPI = ,E12.5
     1,/,1X,6HDPO = ,E12.5,/,1X,6HDAI = ,E12.5
     2,/,1X,6HDAO = ,E12.5)
103   FORMAT(1X,12HERROR IPO = ,I12)
104   FORMAT(1X,12HERROR IAO = ,I12)
      END
```

```
      PROGRAM TESTO

        II=0
        JJ=0

        WRITE(3,198)
198     FORMAT('         IYA IYP IXA IXP IVOA IVOP'
     +,' ZA            ZP            VOA           VOP')
        PI=3.14159265359
        DP=2.0*PI/128.0
        DA=2.0**(1.0/8.0)
        DA2=SQRT(DA)
        ADLMT=DA+0.000001
        PDLMT=DP+0.000001

        ICNT=0
100   CONTINUE
        ICNT=ICNT+1
10      CONTINUE
        IXA=RAN(II,JJ)*'200
        IF(IXA.LT.0.OR.IXA.GT.'177) GO TO 10
20      CONTINUE
        IXP=RAN(II,JJ)*'200
        IF(IXP.LT.0.OR.IXP.GT.'177) GO TO 20
30      CONTINUE
        IYA=RAN(II,JJ)*'200
        IF(IYA.LT.0.OR.IYA.GT.'177) GO TO 30
40      CONTINUE
        IYP=RAN(II,JJ)*'200
        IF(IYP.LT.0.OR.IYP.GT.'177) GO TO 40
        I1VI=IXA*2**8+IXP
        I2VI=IYA*2**8+IYP
C
C DETERMINE LARGEST VECTOR AMPLITUDE
        ILF=IDLVA(I1VI,I2VI)
C NORMALIZE AMPLITUDE
        ITAA=INA(ILF,I1VI,I2VI)
C NORMALIZE PHASE
        ITAP=INP(ILF,I1VI,I2VI,ITAA)
C RANDOMIZE AMPLITUDE
        ITAAR=IRA(ITAP,ITAA)
C AMPLITUDE SIGNIFICANCE LIMIT
        ITAAL=IASL(ITAAR)
C PHASE FOLDOVER QUESTION
        IPFF=IPFQ(ITAP)
C PHASE FOLDOVER
        ITAPF=IPF(IPFF,ITAP)
C NORMALIZE VECTOR ADD TABLE
```

```
      ITDT=INVAT(ITAPF,ITAAL)
C COMBINE EPROM DATA
      ITD=ICED(ITAAL,ITDT)
C SELECT LARGEST VECTOR
      IVIL=ISLV(ILF,I1VI,I2VI)
C INVERSE-NORMALIZE AMPLITUDE
      IVOA=IINA(IVIL,ITD)
C INVERSE-NORMALIZE PHASE & INVERSE PHASE FOLDOVER
      IVOP=IINP(IPFF,IVIL,ITD)
C COMBINE VECTOR OUTPUT
      IVO=ICVO(IVOP,IVOA)
C

      XA=DA**IXA
      YA=DA**IYA

      XP=FLOAT(IXP)
      YP=FLOAT(IYP)
      XP=DP*XP
      YP=DP*YP

      XR=XA*COS(XP)
      XI=XA*SIN(XP)
      YR=YA*COS(YP)
      YI=YA*SIN(YP)

      ZR=XR+YR
      ZI=XI+YI

      ZA=SQRT(ZR**2+ZI**2)/2.0
      ZP=ATAN2(ZI,ZR)

      IVOA=ISWAB(IVOA)
      VOA=DA**IVOA
      VOP=FLOAT(IVOP)
      VOP=DP*VOP
      IF(ZP.LT.0.0.AND.VOP.NE.0.0) ZP=ZP+2.0*PI

      PD=ABS(ZP-VOP)
      AD=ZA/VOA
      IF(AD.LT.1.0) AD=VOA/ZA

      IF(IXA.EQ.IYA.AND.IABS(IXP-IYP).EQ.'100) GO TO 210
      IF(IVOA.EQ.0) GO TO 220

      IF(AD.GT.ADLMT) GO TO 200
      IF(PD.GT.PDLMT) GO TO 200
```

```
      PROGRAM WEIGHT
      DIMENSION IDATA(256)
      PI=3.14159265359
      DA=2.0**(1.0/8.0)
      READ(5,102) RK,NWPNTS
102   FORMAT(1X,F8.4,2(1X,O6))
      READ(2,103) NSTAGE
      NPOINTS=2**NSTAGE
      IF(NSTAGE.LT.1) STOP 'NSTAGE < 1'
      IF(NSTAGE.GT.'10) STOP 'NSTAGE > '10'
      READ(2,103) (IDATA(I),I=1,NPOINTS)
103   FORMAT(1X,O6)
      WRITE(7,102) RK,NWPNTS,NPOINTS
      ISTA=(NPOINTS-NWPNTS)/2
      ISTO=NPOINTS-ISTA+1

      DO 100 I=1,ISTA
100   IDATA(I)='0

      DO 105 I=ISTO,NPOINTS
105   IDATA(I)='0

      ISTO=ISTO-1
      ISTA=ISTA+1

      DPHI=PI/NWPNTS
      PHI=-PI/2.0-DPHI/2.0

      DO 200 I=ISTA,ISTO
      PHI=PHI+DPHI
      W=1.0-RK+RK*COS(PHI)**2
      W=1.0/W
      W=ALOG(W)/ALOG(DA)
      IW=IFIX(W+0.5)
      IW=ISWAB(IW)
      IDATA(I)=IDATA(I)-IW
      IF(IDATA(I).LT.0) IDATA(I)='177.AND.IDATA(I)
200   CONTINUE

      REWIND 2
      WRITE(2,103) NSTAGE
      WRITE(2,103) (IDATA(I),I=1,NPOINTS)

      END
```

```
C BIT REVERSAL SHUFFLE OF DFFT DATA
C
      DIMENSION IDATA(256)
      READ(2,101) NSTAGE
      JPOINTS=2**NSTAGE
      IF(JPOINTS.LT.1) STOP 'JPOINTS < 1'
      IF(JPOINTS.GT.256) STOP 'JPOINTS > 256'
      READ(2,101) (IDATA(J),J=1,JPOINTS)
C
      JJ=-1
      DO 10 J=1,JPOINTS
      JJ=JJ+1
      JBR=IBITR(JJ,NSTAGE)+1
      IF(JBR.LT.J) GO TO 10
      ITMP=IDATA(J)
      IDATA(J)=IDATA(JBR)
      IDATA(JBR)=ITMP
10    CONTINUE
C
      REWIND 2
      WRITE(2,101) NSTAGE
      WRITE(2,101) (IDATA(J),J=1,JPOINTS)
101   FORMAT(1X,06)
      END
```

```
      PROGRAM STAGE
C 15-DEC-76  COKER
C DFFT STAGE IN LOG-POLAR QUANTIZATION
      LOGICAL*1 LTMP(2),LDATA(2,256),LI2VI(2)
      DIMENSION IDATA(256)
      EQUIVALENCE (IDATA(1),LDATA(1,1)),(ITMP,LTMP(1))
     +,(I2VI,LI2VI(1))
C
      READ(2,101) NSTAGE
      JPOINTS=2**NSTAGE
      NS2=NSTAGE-1
      IF(JPOINTS.LT.1) STOP 'JPOINTS < 1'
      IF(JPOINTS.GT.256) STOP 'JPOINTS > 256'
      READ(2,101) (IDATA(J),J=1,JPOINTS)
101   FORMAT(1X,O6)
      WRITE(7,103)
103   FORMAT(1X,'ISIGN,JBLOCK = ? 2(1X,O6)')
      READ(5,106) ISIGN,JBLOCK
106   FORMAT(1X,I6,2(1X,O6))
      IF(IABS(ISIGN).NE.1) STOP 'ISIGN'
      IF(JBLOCK.LT.2) STOP 'JBLOCK < 2'
      IF(JBLOCK.GT.JPOINTS) STOP 'JBLOCK > JPOINTS'
      JOFFSET=JBLOCK/2
      WRITE(7,106) ISIGN,JBLOCK,JOFFSET
C
C POINT BLOCK LOOP
      JJ=0
      JJSTP='200/JPOINTS
      IF(JPOINTS.GT.'200) JJSTP=JPOINTS/'200
      DO 100 JSTA=1,JPOINTS,JBLOCK
      JSTO=JSTA+JOFFSET-1
      JJBR=IBITR(JJ,NS2)
      IF(JPOINTS.LE.'200) JJBR=JJBR*JJSTP
      IF(JPOINTS.GT.'200) JJBR=JJBR/JJSTP
      IF(JJBR.NE.0) JJBR='200-JJBR
C     INVERSE DFFT ?
      IF(ISIGN.EQ.-1.AND.JJBR.NE.0) JJBR='200-JJBR
      WRITE(7,102) JSTA,JJ,JJBR
102   FORMAT(3(1X,O6))
      JJ=JJ+1
C
C POINT LOOP
      DO 100 J=JSTA,JSTO
      ITMP=IDATA(J)
      I1VI=ITMP
      I2VI=IDATA(J+JOFFSET)
C
C DETERMINE LARGEST VECTOR AMPLITUDE
```

```
        ILF=IDLVA(I1VI,I2VI)
C NORMALIZE AMPLITUDE
        ITAA=INA(ILF,I1VI,I2VI)
C NORMALIZE PHASE
        ITAP=INP(ILF,I1VI,I2VI,ITAA)
C RANDOMIZE AMPLITUDE
        ITAAR=IRA(ITAP,ITAA)
C AMPLITUDE SIGNIFICANCE LIMIT
        ITAAL=IASL(ITAAR)
C PHASE FOLDOVER QUESTION
        IPFF=IPFQ(ITAP)
C PHASE FOLDOVER
        ITAPF=IPF(IPFF,ITAP)
C NORMALIZE VECTOR ADD TABLE
        ITDT=INVAT(ITAPF,ITAAL)
C COMBINE EPROM DATA
        ITD=ICED(ITAAL,ITDT)
C SELECT LARGEST VECTOR
        IVIL=ISLV(ILF,I1VI,I2VI)
C INVERSE-NORMALIZE AMPLITUDE
        IVOA=IINA(IVIL,ITD)
C INVERSE-NORMALIZE PHASE & INVERSE PHASE FOLDOVER
        IVOP=IINP(IPFF,IVIL,ITD)
C COMBINE VECTOR OUTPUT
        IVO=ICVO(IVOP,IVOA)
        IDATA(J)=IVO
C     180 DEG VECTOR ROTATE - MINUS VECTOR
        I2VI=IDATA(J+JOFFSET)
        LI2VI(1)=LI2VI(1)+'100
C     CLEAR 8TH BIT
        I2VI=I8CLR(I2VI)
        I1VI=ITMP
C
C DETERMINE LARGEST VECTOR AMPLITUDE
        ILF=IDLVA(I1VI,I2VI)
C NORMALIZE AMPLITUDE
        ITAA=INA(ILF,I1VI,I2VI)
C NORMALIZE PHASE
        ITAP=INP(ILF,I1VI,I2VI,ITAA)
C RANDOMIZE AMPLITUDE
        ITAAR=IRA(ITAP,ITAA)
C AMPLITUDE SIGNIFICANCE LIMIT
        ITAAL=IASL(ITAAR)
C PHASE FOLDOVER QUESTION
        IPFF=IPFQ(ITAP)
C PHASE FOLDOVER
        ITAPF=IPF(IPFF,ITAP)
C NORMALIZE VECTOR ADD TABLE
```

```
        ITDT=INVAT(ITAPF,ITAAL)
C COMBINE EPROM DATA
        ITD=ICED(ITAAL,ITDT)
C SELECT LARGEST VECTOR
        IVIL=ISLV(ILF,I1VI,I2VI)
C INVERSE-NORMALIZE AMPLITUDE
        IVOA=IINA(IVIL,ITD)
C INVERSE-NORMALIZE PHASE & INVERSE PHASE FOLDOVER
        IVOP=IINP(IPFF,IVIL,ITD)
C COMBINE VECTOR OUTPUT
        IVO=ICVO(IVOP,IVOA)
        ITMP=IVO
C     FORIER PHASE WEIGHT
        LTMP(1)=LTMP(1)+JJBR
C     CLEAR 8TH BIT
        IDATA(J+JOFFSET)=I8CLR(ITMP)
        IF(ISIGN.NE.-1) GO TO 95
        IDATA(J)=IDATA(J)+"4000
        IDATA(J+JOFFSET)=IDATA(J+JOFFSET)+"4000
95      CONTINUE
100     CONTINUE
C
        REWIND 2
        WRITE(2,101) NSTAGE
        WRITE(2,101) (IDATA(J),J=1,JPOINTS)
        WRITE(3,104) NSTAGE
104     FORMAT(////,1X,03,/)
        WRITE(3,105) (LDATA(1,J),LDATA(2,J),J=1,JPOINTS)
105     FORMAT(2(3X,03))
        END
```

```
$JOB/RT11
.R PIP
*TT:=FTN2.DAT
*TMP=FTN2.DAT
.RUN SHUF
.R PIP
*TT:=FTN2.DAT
*TMP=TMP,FF,FTN2.DAT
$RUN STAGE
$DATA
      1       2
$EOD
$RT11
.R PIP
*TT:=FTN3.DAT
*TMP=TMP,FF,FTN3.DAT
$RUN STAGE
$DATA
      1       4
$EOD
$RT11
.R PIP
*TT:=FTN3.DAT
*TMP=TMP,FF,FTN3.DAT
$RUN STAGE
$DATA
      1      10
$EOD
$RT11
.R PIP
*TT:=FTN3.DAT
*TMP=TMP,FF,FTN3.DAT
$RUN STAGE
$DATA
      1      20
$EOD
$RT11
.R PIP
*TT:=FTN3.DAT
*TMP=TMP,FF,FTN3.DAT
$RUN STAGE
$DATA
      1      40
$EOD
$RT11
.R PIP
*TT:=FTN3.DAT
*TMP=TMP,FF,FTN3.DAT
$EOJ
```

```
      PROGRAM TESTVA
100   CONTINUE
      WRITE(7,101)
101   FORMAT(1X,1H?)
      READ(5,102) ISW
102   FORMAT(I2)
      IF(ISW.GT.3) GO TO 999
      GO TO (1,2,3) ISW
1     CONTINUE
      READ(5,103) IX1A,IX1P,IX2A,IX2P
103   FORMAT(6(1X,03))
      WRITE(7,103) IX1A,IX1P,IX2A,IX2P
      IX1=IX1A*2**8+IX1P
      IX2=IX2A*2**8+IX2P
      GO TO 100
2     CONTINUE
C
        I1VI=IX1
        I2VI=IX2

        CALL IPOKE('167762,I1VI)
        CALL IPOKE('167752,I2VI)

C DETERMINE LARGEST VECTOR AMPLITUDE
        ILF=IDLVA(I1VI,I2VI)
C NORMALIZE AMPLITUDE
        ITAA=INA(ILF,I1VI,I2VI)
C NORMALIZE PHASE
        ITAP=INP(ILF,I1VI,I2VI,ITAA)
C RANDOMIZE AMPLITUDE
        ITAAR=IRA(ITAP,ITAA)
C AMPLITUDE SIGNIFICANCE LIMIT
        ITAAL=IASL(ITAAR)
C PHASE FOLDOVER QUESTION
        IPFF=IPFQ(ITAP)
C PHASE FOLDOVER
        ITAPF=IPF(IPFF,ITAP)
C NORMALIZE VECTOR ADD TABLE
        ITDT=INVAT(ITAPF,ITAAL)
C COMBINE EPROM DATA
        ITD=ICED(ITAAL,ITDT)
C SELECT LARGEST VECTOR
        IVIL=ISLV(ILF,I1VI,I2VI)
C INVERSE-NORMALIZE AMPLITUDE
        IVOA=IINA(IVIL,ITD)
C INVERSE-NORMALIZE PHASE & INVERSE PHASE FOLDOVER
        IVOP=IINP(IPFF,IVIL,ITD)
C COMBINE VECTOR OUTPUT
```

```
        IVO=ICVO(IVOP,IVOA)
        IY=IVO
C
        IVOH=IPEEK('167764)

      IYA=IY/2**8
      IYP=IY-IYA*2**8
3     CONTINUE
      WRITE(7,103) IX1A,IX1P,IX2A,IX2P,IYA,IYP
      WRITE(7,104) IY,IVOH
104   FORMAT(2(1X,O6))
C
        WRITE(7,105) ILF,ITAA,ITAAL,ITAP,IPFF,ITAPF
     +,ITDT,ITD,IVIL,IVOA,IVOP,IVO
105     FORMAT(12(1X,O6))
C
      GO TO 100
999   CONTINUE
      END
```

```
      PROGRAM TESTH
        I=0
        J=0
        K=0

        II=0
        JJ=0

        PI=3.14159265359
        DP=2.0*PI/128.0
        DA=2.0**(1.0/8.0)
        DA2=SQRT(DA)
        ADLMT=DA+0.000001
        PDLMT=DP+0.000001

100   CONTINUE
        IF(I.EQ.256) I=0
        ICNT=ICNT+1
10      CONTINUE
        IXA=RAN(II,JJ)*'200
        IF(IXA.LT.0.OR.IXA.GT.'177) GO TO 10
20      CONTINUE
        IXP=RAN(II,JJ)*'200
        IF(IXP.LT.0.OR.IXP.GT.'177) GO TO 20
30      CONTINUE
        IYA=RAN(II,JJ)*'200
        IF(IYA.LT.0.OR.IYA.GT.'177) GO TO 30
40      CONTINUE
        IYP=RAN(II,JJ)*'200
        IF(IYP.LT.0.OR.IYP.GT.'177) GO TO 40
        I1VI=IXA*2**8+IXP
        I2VI=IYA*2**8+IYP
        CALL IPOKE('167762,I1VI)
        CALL IPOKE('167752,I2VI)
C
C DETERMINE LARGEST VECTOR AMPLITUDE
        ILF=IDLVA(I1VI,I2VI)
C NORMALIZE AMPLITUDE
        ITAA=INA(ILF,I1VI,I2VI)
C NORMALIZE PHASE
        ITAP=INP(ILF,I1VI,I2VI,ITAA)
C RANDOMIZE AMPLITUDE
        ITAAR=IRA(ITAP,ITAA)
C AMPLITUDE SIGNIFICANCE LIMIT
        ITAAL=IASL(ITAAR)
C PHASE FOLDOVER QUESTION
        IPFF=IPFQ(ITAP)
C PHASE FOLDOVER
```

```
      ITAPF=IPF(IPFF,ITAP)
C NORMALIZE VECTOR ADD TABLE
      ITDT=INVAT(ITAPF,ITAAL)
C COMBINE EPROM DATA
      ITD=ICED(ITAAL,ITDT)
C SELECT LARGEST VECTOR
      IVIL=ISLV(ILF,I1VI,I2VI)
C INVERSE-NORMALIZE AMPLITUDE
      IVOA=IINA(IVIL,ITD)
C INVERSE-NORMALIZE PHASE & INVERSE PHASE FOLDOVER
      IVOP=IINP(IPFF,IVIL,ITD)
C COMBINE VECTOR OUTPUT
      IVOC=ICVO(IVOP,IVOA)
C
      IVO=IPEEK('167764)

      IF(I.NE.0) GO TO 200
      IF(J.EQ.32767) GO TO 900
      J=J+1
      WRITE(7,998) K,J
998   FORMAT(1X'START BLOCK = ',I6,1X,I6)

200   CONTINUE
      I=I+1
      IF(IVO.EQ.IVOC) GO TO 100
      WRITE(7,999) I,I1VI,I2VI,IVO,IVOC
999   FORMAT(1X'ERROR',7(1X,O6))
      WRITE(7,999) ILF,ITAA,ITAP,ITAAR,ITAAL,IPFF,ITAPF
      WRITE(7,999) ITDT,ITD,IVIL,IVOA,IVOP
      PAUSE
      GO TO 100
900   CONTINUE
      J=0
      K=K+1
      GO TO 200

      END
```

```
        .TITLE IDLVA  DETERMINE LARGEST VECTOR AMPLITUDE
; ILF=IDLVA(I1VI,I2VI)

        .MCALL  ..V2..  .REGDEF
        ..V2..
        .REGDEF
        .GLOBL  IDLVA

IDLVA:  MOV     (R5)+   ,R0     ;GET # OF ARGUMENTS
        CMPB    #2      ,R0     ;2 ARG ?
        BNE     ERR             ;IF NOT THEN ERROR

;**** LOAD GENERAL REGISTERS
        MOV     @(R5)+  ,R1     ;LOAD I1VI
        MOV     @(R5)   ,R2     ;LOAD I2VI

        BIC     #100377 ,R1     ;CLEAR PHASE BITS & (15)
        BIC     #100377 ,R2     ;CLEAR PHASE BITS & (15)
        MOV     #0      ,R0     ;ILF=0 I1VI IS LARGEST
        CMP     R1      ,R2     ;I1VI-I2VI
        BGT     SKIP1           ;IF I1VI LARGER THEN SKIP
        MOV     #1      ,R0     ;ILF=1 I2VI IS > OR =

SKIP1:  RTS     PC              ;RETURN

;**** ERROR SECTION
ERR:    JMP     @#4             ;ABORT TO MONITOR

        .END
```

```
        .TITLE  INA  NORMALIZE AMPLITUDE 26-AUG-77 COKER
; ITAA=INA(ILF,I1VI,I2VI)

        .MCALL  ..V2..  .REGDEF
        ..V2..
        .REGDEF
        .GLOBL  INA

INA:    MOV     (R5)+   ,R0     ;GET # OF ARGUMENTS
        CMPB    #3      ,R0     ;3 ARG ?
        BNE     ERR             ;IF NOT THEN ERROR

;**** LOAD GENERAL REGISTERS
        MOV     @(R5)+  ,R3     ;LOAD ILF
        MOV     @(R5)+  ,R1     ;LOAD I1VI
        MOV     @(R5)   ,R2     ;LOAD I2VI

        BIC     #100377 ,R1     ;CLEAR PHASE BITS & (15)
        BIC     #100377 ,R2     ;CLEAR PHASE BITS & (15)

;**** -I2VI M2 OR -I1VI M1 ??
        CMP     #0      ,R3     ;ILF=0?
        BEQ     M2              ;IF SO JMP -I2VI
        CMP     #1      ,R3     ;ILF=1?
        BEQ     M1              ;IF SO JMP -I1VI

;**** ERROR SECTION
ERR:    JMP     @#4             ;ABORT TO MONITOR

;**** M2
M2:     COM     R2              ;COMPLEMENT I2VI
        BIC     #377    ,R2     ;CLEAR PHASE BITS
        ADD     #400    ,R2     ;-I2VI
        JMP     ADD             ;GO TO ADDITION

;**** M1
M1:     COM     R1              ;COMPLEMENT I1VI
        BIC     #377    ,R1     ;CLEAR PHASE BITS
        ADD     #400    ,R1     ;-I1VI
        JMP     ADD             ;GO TO ADDITION

;**** ADDITION
ADD:    ADD     R1      ,R2     ;IABS(I1VI-I2VI)
        MOV     R2      ,R0
        BIC     #100000 ,R0     ;CLEAR BIT (15)
        RTS     PC              ;RETURN

        .END
```

```
        .TITLE  INP  NORMALIZE PHASE
; ITAP=INP(ILF,I1VI,I2VI,ITAA)

        .MCALL  ..V2..  .REGDEF
        ..V2..
        .REGDEF
        .GLOBL  INP

INP:    MOV     (R5)+   ,R0     ;GET # OF ARGUMENTS
        CMPB    #4      ,R0     ;4 ARG ?
        BNE     ERR             ;IF NOT THEN ERROR

;**** LOAD GENERAL REGISTERS
        MOV     @(R5)+  ,R3     ;LOAD ILF
        MOV     @(R5)+  ,R1     ;LOAD I1VI
        MOV     @(R5)+  ,R2     ;LOAD I2VI
        MOV     @(R5)   ,R4     ;LOAD ITAA

        BIC     #177600 ,R1     ;CLEAR AMPLITUDE BITS & (07)
        BIC     #177600 ,R2     ;CLEAR AMPLITUDE BITS & (07)
        SWAB    R4              ;ITAA TO PHASE BITS
        BIC     #177776 ,R4     ;LSB ONLY OF ITAA

;**** -I2VI M2 OR -I1VI M1 ??
        CMP     #0      ,R3     ;ILF=0?
        BEQ     M2              ;IF SO -I2VI
        CMP     #1      ,R3     ;ILF=1?
        BEQ     M1              ;IF SO -I1VI

;**** ERROR SECTION
ERR:    JMP     @#4             ;ABORT TO MONITOR

;**** M2
M2:     COM     R2              ;COMPLEMENT I2VI
        ADD     R4      ,R2     ; RND +1 OR 0 = (0 OR -1)
        BIC     #177400 ,R2     ;CLEAR AMPLITUDE BITS
        JMP     ADD             ;GO TO ADD

;**** M1
M1:     COM     R1              ;COMPLEMENT I1VI
        ADD     R4      ,R1     ; RND +1 OR 0 = (0 OR -1)
        BIC     #177400 ,R1     ;CLEAR AMPLITUDE BITS
        JMP     ADD             ;GO TO ADD

;**** ADD
ADD:    ADD     R1      ,R2     ;IABS(I1VI-I2VI)
        MOV     R2      ,R0
        BIC     #177600 ,R0     ;LIMIT TO LOWER 7 BITS

        RTS     PC              ;RETURN

        .END
```

```
        .TITLE IRA RANDOMIZE AMPLITUDE 9-SEP-77 COKER
; ITAAR=IRA(ITAP,ITAA)

        .MCALL  ..V2..  .REGDEF
        ..V2..
        .REGDEF
        .GLOBL  IRA

IRA:    MOV     (R5)+   ,R0     ;GET # OF ARGUMENTS
        CMPB    #2      ,R0     ;2 ARG ?
        BNE     ERR             ;IF NOT THEN ERROR

;**** LOAD GENERAL REGISTERS
        MOV     @(R5)+  ,R1     ;LOAD ITAP
        MOV     @(R5)   ,R0     ;LOAD ITAA

        BIC     #177776 ,R1     ;LSB OF ITAP ONLY
        SWAB    R1              ;MOVE TO AMPLITUDE
        ADD     R1      ,R0     ;RANDONIZE PHASE

        RTS     PC              ;RETURN

;**** ERROR SECTION
ERR:    JMP     @#4             ;ABORT TO MONITOR

        .END
```

```
        .TITLE  IASL  AMPLITUDE SIGNIFICANCE LIMIT
; ITAAL=IASL(ITAAR)

        .MCALL  ..V2..  .REGDEF
        ..V2..
        .REGDEF
        .GLOBL  IASL

IASL:   MOV     (R5)+   ,R0     ;GET # OF ARGUMENTS
        CMPB    #1      ,R0     ;1 ARG ?
        BNE     ERR             ;IF NOT THEN ERROR

;**** LOAD GENERAL REGISTERS
        MOV     @(R5)   ,R1     ;LOAD ITAAR

;**** ALF=?
        MOV     #0      ,R3     ;ALF=0
        MOV     R1      ,R0
        BIC     #147777 ,R0     ;TA12 & TA13 ONLY
        CMP     #30000  ,R0     ;TA12=1 & TA13=1 ?
        BNE     SKIP1           ;IF NOT THEN SKIP
        MOV     #1      ,R3     ;ALF=1
SKIP1:  MOV     R1      ,R0
        BIC     #137777 ,R0     ;TA14=1?
        BEQ     SKIP2           ;IF NOT THEN SKIP
        MOV     #1      ,R3     ;ALF=1

;**** IF AFL=1 THEN SET TA10-13
SKIP2:  MOV     R1      ,R0
        CMP     #0      ,R3     ;AFL=0?
        BEQ     SKIP3           ;IF SO JMP TO RETURN
        BIS     #26000  ,R0     ;SET TA10,11,&13
        BIC     #10000  ,R0     ;CLEAR TA12

SKIP3:  BIC     #140377 ,R0     ;CLEAR PHASE, (14), & (15)
        RTS     PC              ;RETURN

;**** ERROR SECTION
ERR:    JMP     @#4             ;ABORT TO MONITOR

        .END
```

```
        .TITLE  IPFQ  PHASE FOLDOVER QESTION 26-AUG-77 COKER
; IPFF=IPFQ(ITAP)

        .MCALL  ..V2..  .REGDEF
        ..V2..
        .REGDEF
        .GLOBL  IPFQ

IPFQ:   MOV     (R5)+   ,R0     ;GET # OF ARGUMENTS
        CMPB    #1      ,R0     ;1 ARG ?
        BNE     ERR             ;IF NOT THEN ERROR

;**** LOAD GENERAL REGISTERS
        MOV     @(R5)   ,R1     ;LOAD ITAP

        MOV     #0      ,R0     ;IPFF=0
        BIT     #100    ,R1     ;(06) BIT ONLY
        BNE     SKIP1           ;IF SET THEN SKIP
        MOV     #1      ,R0     ;IPFF=1
SKIP1:  RTS     PC              ;RETURN

;**** ERROR SECTION
ERR:    JMP     @#4             ;ABORT TO MONITOR

        .END
```

```
        .TITLE  I PHASE FOLDOVER 26-AUG-77 COKER
; ITAPF=IPF(IPFF,ITAP)

        .MCALL  ..V2..  .REGDEF
        ..V2..
        .REGDEF
        .GLOBL  IPF

IPF:    MOV     (R5)+   ,R0     ;GET # OF ARGUMENTS
        CMPB    #2      ,R0     ;2 ARG ?
        BNE     ERR             ;IF NOT THEN ERROR

;**** LOAD GENERAL REGISTERS
        MOV     @(R5)+  ,R1     ;LOAD IPFF
        MOV     @(R5)   ,R0     ;LOAD ITAP

;**** PHASE FOLDOVER ?
        CMP     #0      ,R1     ;IPFF=0?
        BNE     SKIP1           ;IF NOT THEN SKIP
        COM     R0              ;FOLD ITAP
        BIC     #177700 ,R0     ;6 LOWER BITS ONLY
RET:    RTS     PC              ;RETURN
SKIP1:  CMP     #1      ,R1     ;IPFF=1?
        BEQ     RET             ;IF SO RETURN

;**** ERROR SECTION
ERR:    JMP     @#4             ;ABORT TO MONITOR

        .END
```

```
        .TITLE  INVAT  NORMALIZE VECTOR ADD TABLE
; ITDT=INVAT(ITAPF,ITAAL)

        .MCALL  ..V2..  .REGDEF
        ..V2..
        .REGDEF
        .GLOBL  INVAT,  EPROM1, EPROM2

INVAT:  MOV     (R5)+   ,R0     ;GET # OF ARGUMENTS
        CMPB    #2      ,R0     ;2 ARG ?
        BNE     ERF             ;IF NOT THEN ERROR

;**** LOAD GENERAL REGISTERS
        MOV     @(R5)+  ,R1     ;LOAD ITAPF
        MOV     @(R5)   ,R2     ;LOAD ITAAL

;**** FIRST 9 BITS OF EPROM ADDRESS
        MOV     R2      ,R3     ;UPPER PART
        BIC     #377    ,R3     ;CLEAR PHASE OF ITAAL
        BIC     #177400 ,R1     ;CLEAR AMP OF ITAPF
        BIS     R1      ,R3     ;LOWER PART
        ASLB    R3              ;COMPRESS
        ASLB    R3              ;COMPRESS
        ASR     R3              ;ALIGN
        ASR     R3              ;ALIGN
        BIC     #176000 ,R3     ;ONLY TA00-05 & TA08-11
        MOV     #0      ,R0     ;ZERO EPROM DATA

;**** READ EPROM2 ?
        BIS     #177400 ,R0     ;SET BITS IF ROM NOT SELETED
        BIT     #10000  ,R2     ;TA12=0?
        BNE     SKIP3           ;IF NOT THEN SKIP EPROM2
        BIT     #20000  ,R2     ;TA13=1?
        BEQ     SKIP4           ;IF NOT THEN SKIP
        BIS     #2000   ,R3     ;ADDRESS 2ED K
SKIP4:  MOVB    EPROM2(R3) ,R0  ;GET EPROM2.DATA
        SWAB    R0              ;EPROM2 DATA IN UPPER BYTE
        BIC     #377    ,R0     ;CLEAR LOWER BYTE

;**** READ EPROM1 ?
SKIP3:  BIS     #377    ,R0     ;SET BITS IF ROM NOT SELETED
        BIC     #2000   ,R3     ;RESET TO 1K ADDRESS
        BIT     #20000  ,R2     ;TA13=0?
        BNE     SKIP1           ;IF NOT THEN SKIP EPROM1
        BIT     #10000  ,R2     ;TA12=1?
        BEQ     SKIP2           ;IF NOT THEN SKIP
        BIS     #2000   ,R3     ;ADDRESS 2ED. K
SKIP2:  MOVB    EPROM1(R3) ,R4  ;GET EPROM1 DATA
```

```
        BIC     #377    ,R0
        BIC     #177400 ,R4     ;CLEAR UPPER BYTE
        BIS     R4      ,R0     ;EPROM1 IN LOWER BYTE

SKIP1:  RTS     PC              ;RETURN

;**** ERROR SECTION
ERR:    JMP     @#4             ;ABORT TO MONITOR

        .END
```

```
        .TITLE ICED  COMBINE EPROM DATA 26-AUG-77 COKER
; ITD=ICED(ITAAL,ITDT)

        .MCALL  ..V2..  .REGDEF
        ..V2..
        .REGDEF
        .GLOBL  ICED

ICED:   MOV     (R5)+   ,R0     ;GET # OF ARGUMENTS
        CMPB    #2      ,R0     ;2 ARG ?
        BNE     ERR             ;IF NOT THEN ERROR

;**** LOAD GENERAL REGISTERS
        MOV     @(R5)+  ,R1     ;LOAD ITAAL
        MOV     @(R5)   ,R2     ;LOAD ITDT

;**** TA12 & TA13
        MOV     #0      ,R3
        BIT     #10000  ,R1     ;TA12=1?
        BEQ     SKIP1           ;IF NOT SKIP
        BIS     #30060  ,R3     ;TA12
SKIP1:  BIT     #20000  ,R1     ;TA13=1?
        BEQ     SKIP2           ;IF NOT SKIP
        BIS     #16     ,R3     ;TA12+TA13

;**** EPROM1 DATA
SKIP2:  MOVB    R2      ,R4     ;EPROM1 DATA
        ASL     R4
        ASL     R4
        ASL     R4
        ASL     R4
        ASRB    R4
        ASRB    R4
        ASRB    R4              ;1-4 > TA00-03 &
        ASRB    R4              ;5-8 > TA08-11
        BIC     #170360 ,R4     ;CLEAR TA04-07 & TA12-15

;**** EPROM2 DATA
        MOV     R4      ,R0     ;OUTPUT
        BIC     #7401   ,R0     ;CLEAR ANDED BITS
        BIT     #400    ,R2     ;BIT1=1?
        BEQ     SK1
        BIT     #1      ,R4
        BEQ     SK1
        BIS     #1      ,R0
SK1:    BIT     #1000   ,R2     ;BIT2=1?
        BEQ     SK2
        BIT     #400    ,R4
```

```
        BEQ     SK2
        BIS     #400    ,R0
SK2:    BIT     #2000   ,R2     ;BIT3=1?
        BEQ     SK9
        BIT     #1000   ,R4
        BEQ     SK3
        BIS     #1000   ,R0
SK3:    BIT     #2000   ,R4
        BEQ     SK9
        BIS     #2000   ,R0
SK9:    BIT     #4000   ,R2     ;BIT4=1?
        BEQ     SK4
        BIT     #4000   ,R4
        BEQ     SK4
        BIS     #4000   ,R0
SK4:    BIT     #10000  ,R2     ;BIT5=1?
        BEQ     SK5
        BIS     #20     ,R0
SK5:    BIT     #20000  ,R2     ;BIT6=1?
        BEQ     SK6
        BIS     #40     ,R0
SK6:    BIT     #40000  ,R2     ;BIT7=1?
        BEQ     SK7
        BIS     #10000  ,R0
SK7:    BIT     #100000 ,R2     ;BIT8=1?
        BEQ     SK8
        BIS     #20000  ,R0
SK8:

;**** COMBINE
        BIS     R3      ,R0     ;+TA12+TA13

        RTS     PC              ;RETURN


;**** ERROR SECTION
ERR:    JMP     @#4             ;ABORT TO MONITOR

        .END
```

```
        .TITLE ISLV  SELECT LARGEST VECTOR 26-AUG-77 COKER
; IVIL=ISLV(ILF,I1VI,I2VI)

        .MCALL  ..V2..  .REGDEF
        ..V2..
        .REGDEF
        .GLOBL  ISLV

ISLV:   MOV     (R5)+   ,R0     ;GET # OF ARGUMENTS
        CMPB    #3      ,R0     ;3 ARG ?
        BNE     ERR             ;IF NOT THEN ERROR

;**** LOAD GENERAL REGISTERS
        MOV     @(R5)+  ,R3     ;LOAD ILF
        MOV     @(R5)+  ,R1     ;LOAD I1VI
        MOV     @(R5)   ,R2     ;LOAD I2VI

        MOV     R1      ,R0     ;I1VI
        BIT     #1      ,R3     ;ILF=1?
        BEQ     SKIP1           ;IF NOT SKIP
        MOV     R2      ,R0     ;I2VI
SKIP1:  RTS     PC              ;RETURN

;**** ERROR SECTION
ERR:    JMP     @#4             ;ABORT TO MONITOR

        .END
```

```
        .TITLE  IINA  INVERSE-NORMALIZE AMPLITUDE
; IVOA=IINA(IVIL,ITD)

        .MCALL  ..V2..  .REGDEF
        ..V2..
        .REGDEF
        .GLOBL  IINA

IINA:   MOV     (R5)+   ,R0     ;GET # OF ARGUMENTS
        CMPB    #2      ,R0     ;2 ARG ?
        BNE     ERR             ;IF NOT THEN ERROR

;**** LOAD GENERAL REGISTERS
        MOV     @(R5)+  ,R0     ;LOAD IVIL
        MOV     @(R5)   ,R1     ;LOAD ITD

        BIC     #377    ,R0     ;CLEAR PHASE
        BIC     #377    ,R1     ;CLEAR PHASE
        BIC     #100000 ,R0     ;CLEAR (15) IVIL
        BIS     #140000 ,R1     ;SET (14)&(15) ITD

        ADD     #400    ,R0     ;CARRY INPUT =1
        ADD     R1      ,R0
        BIT     #100000 ,R0     ;I(UFF)=1?
        BEQ     SKIP1           ;IF NOT SKIP
        BIC     #77400  ,R0     ;IVOA=0
SKIP1:  BIC     #100000 ,R0     ;CLEAR (15)

        RTS     PC              ;RETURN

;**** ERROR SECTION
ERR:    JMP     @#4             ;ABORT TO MONITOR

        .END
```

```
        .TITLE  IINP  INVERSE-NORMALIZE PHASE
; IVOF=IINP(IPFF,IVIL,ITD)

        .MCALL  ..V2..  .REGDEF
        ..V2..
        .REGDEF
        .GLOBL  IINP

IINP:   MOV     (R5)+   ,R0     ;GET # OF ARGUMENTS
        CMPB    #3      ,R0     ;3 ARG ?
        BNE     ERR             ;IF NOT THEN ERROR

;**** LOAD GENERAL REGISTERS
        MOV     @(R5)+  ,R3     ;LOAD IPFF
        MOV     @(R5)+  ,R0     ;LOAD IVIL
        MOV     @(R5)   ,R1     ;LOAD ITD

        BIS     #100    ,R1     ;SET (06) BIT OF PHASE
        ADD     #1      ,R1     ;PHASE OFFSET CORRECTION

;**** PHASE FOLDOVER ?
        BIT     #1      ,R3     ;IPFF=0?
        BNE     SKIP1           ;IF NOT SKIP
        COM     R1              ;COMPLEMENT ITD
        ADD     #1      ,R1     ;-ITD
SKIP1:  BIC     #177600 ,R1     ;CLEAR AMPLITUDE & (7)

;**** ADD
        BIC     #177600 ,R0     ;CLEAR AMPLITUDE & (7)
        ADD     R1      ,R0
        BIC     #177600 ,R0     ;CLEAR AMPLITUDE & (7)

        RTS PC

;**** ERROR SECTION
ERR:    JMP     @#4             ;ABORT TO MONITOR

        .END
```

```
        .TITLE ICVO  COMBINE VECTOR OUTPUT 26-AUG-77 COKER
; IVO=ICVO(IVOP,IVOA)

        .MCALL  ..V2..  .REGDEF
        ..V2..
        .REGDEF
        .GLOBL  ICVO

ICVO:   MOV     (R5)+   ,R0     ;GET # OF ARGUMENTS
        CMPB    #2      ,R0     ;2 ARG ?
        BNE     ERR             ;IF NOT THEN ERROR

;**** LOAD GENERAL REGISTERS
        MOV     @(R5)+  ,R1     ;LOAD IVOP
        MOV     @(R5)   ,R0     ;LOAD IVOA

        BIC     #377    ,R0     ;CLEAR PHASE OF IVOA
        BIC     #177400 ,R1     ;CLEAR AMP OF IVOP
        BIS     R1      ,R0     ;COMBINE IVOA & IVOP

        RTS     PC              ;RETURN

;**** ERROR SECTION
ERR:    JMP     @#4             ;ABORT TO MONITOR

        .END
```

```
        .TITLE EPROM1 9-SEP-77 COKER
        .GLOBL  EPROM1

EPROM1:
.BYTE   377,  376,  376,  375,  375,  374,  374,  373
.BYTE   373,  372,  372,  371,  371,  370,  370,  347
.BYTE   347,  346,  346,  345,  345,  344,  323,  323
.BYTE   322,  322,  321,  301,  300,  300,  317,  277
.BYTE   276,  276,  275,  255,  254,  234,  233,  233
.BYTE   212,  212,  171,  170,  150,  147,  127,  126
.BYTE   106,   65,   44,   24,   23,    3,  342,  321
.BYTE   300,  240,  217,  155,   74,   11,  265,  132

.BYTE   377,  376,  376,  375,  375,  374,  374,  373
.BYTE   353,  352,  352,  351,  351,  350,  350,  347
.BYTE   347,  346,  346,  325,  325,  325,  324,  324
.BYTE   323,  303,  302,  302,  301,  261,  260,  260
.BYTE   277,  257,  256,  256,  235,  235,  234,  214
.BYTE   213,  173,  173,  152,  152,  131,  131,  110
.BYTE    70,   67,   47,   27,    6,  366,  346,  325
.BYTE   265,  245,  205,  145,   66,  367,  253,  105

.BYTE   357,  356,  356,  355,  355,  354,  354,  353
.BYTE   353,  353,  352,  352,  351,  351,  350,  330
.BYTE   327,  327,  326,  326,  325,  325,  305,  304
.BYTE   304,  303,  303,  262,  262,  261,  261,  241
.BYTE   240,  240,  257,  237,  236,  236,  216,  215
.BYTE   175,  174,  154,  154,  133,  133,  113,  112
.BYTE    72,   52,   52,   31,   11,  371,  351,  331
.BYTE   272,  252,  213,  174,  136,   41,    5,  373

.BYTE   357,  356,  356,  355,  355,  355,  354,  354
.BYTE   353,  353,  332,  332,  331,  331,  331,  330
.BYTE   330,  327,  327,  306,  306,  306,  305,  305
.BYTE   304,  264,  263,  263,  263,  262,  242,  241
.BYTE   241,  221,  220,  220,  217,  217,  217,  176
.BYTE   176,  176,  155,  155,  135,  135,  114,   74
.BYTE    74,   54,   34,   34,   14,  374,  354,  335
.BYTE   315,  276,  237,  201,  163,  146,  131,  115

.BYTE   337,  336,  336,  336,  335,  335,  334,  334
.BYTE   333,  333,  333,  332,  332,  331,  331,  330
.BYTE   310,  310,  307,  307,  306,  306,  306,  265
.BYTE   265,  264,  264,  264,  243,  243,  243,  242
.BYTE   222,  222,  221,  201,  201,  200,  160,  160
.BYTE   177,  157,  157,  137,  137,  116,  116,   76
.BYTE    76,   56,   36,   36,   17,  377,  357,  340
.BYTE   321,  302,  263,  244,  226,  210,  213,  216
```

```
.BYTE   337,  336,  336,  336,  335,  335,  334,  334
.BYTE   334,  333,  333,  332,  312,  312,  311,  311
.BYTE   310,  310,  310,  307,  307,  267,  266,  266
.BYTE   265,  265,  245,  244,  244,  244,  243,  223
.BYTE   223,  222,  202,  202,  202,  201,  161,  161
.BYTE   141,  140,  140,  120,  120,  100,  100,   60
.BYTE    60,   40,   40,   20,    1,    1,  362,  342
.BYTE   343,  324,  305,  307,  270,  252,  254,  256

.BYTE   337,  336,  336,  336,  335,  315,  315,  314
.BYTE   314,  313,  313,  313,  312,  312,  312,  311
.BYTE   311,  310,  270,  270,  267,  267,  267,  266
.BYTE   246,  246,  245,  245,  245,  224,  224,  224
.BYTE   224,  203,  203,  203,  203,  162,  162,  162
.BYTE   142,  142,  142,  121,  121,  101,  101,   61
.BYTE    62,   42,   42,   22,   23,    3,    4,  364
.BYTE   345,  346,  327,  330,  312,  313,  315,  316

.BYTE   317,  316,  316,  316,  315,  315,  315,  314
.BYTE   314,  314,  313,  313,  313,  312,  312,  271
.BYTE   271,  271,  270,  270,  270,  267,  267,  247
.BYTE   247,  246,  246,  246,  225,  225,  225,  225
.BYTE   204,  204,  204,  204,  163,  163,  163,  143
.BYTE   143,  143,  123,  123,  123,  103,  103,   63
.BYTE    63,   63,   43,   44,   24,   25,    5,    6
.BYTE   367,  367,  350,  351,  353,  354,  335,  336

.BYTE   317,  316,  316,  316,  315,  315,  315,  314
.BYTE   314,  314,  313,  273,  273,  272,  272,  272
.BYTE   271,  271,  271,  271,  250,  250,  250,  247
.BYTE   247,  247,  247,  226,  226,  226,  226,  225
.BYTE   205,  205,  205,  205,  164,  164,  164,  144
.BYTE   144,  144,  124,  124,  124,  104,  104,  104
.BYTE    64,   65,   45,   45,   46,   26,   27,    7
.BYTE    10,   11,  371,  372,  373,  374,  375,  356

.BYTE   317,  317,  276,  276,  276,  275,  275,  275
.BYTE   274,  274,  274,  273,  273,  273,  272,  272
.BYTE   272,  272,  251,  251,  251,  250,  250,  250
.BYTE   250,  227,  227,  227,  227,  226,  226,  206
.BYTE   206,  206,  205,  165,  165,  165,  165,  145
.BYTE   145,  145,  125,  125,  125,  105,  105,  105
.BYTE    65,   66,   66,   46,   47,   47,   30,   30
.BYTE    31,   12,   12,   13,   14,   15,   16,  377

.BYTE   277,  277,  276,  276,  276,  275,  275,  275
.BYTE   274,  274,  274,  274,  273,  273,  273,  252
```

```
.BYTE   252,  252,  252,  251,  251,  251,  251,  250
.BYTE   230,  230,  230,  227,  227,  227,  207,  207
.BYTE   206,  206,  166,  166,  166,  166,  166,  146
.BYTE   146,  146,  126,  126,  126,  126,  106,  106
.BYTE   106,   67,   67,   67,   50,   50,   51,   51
.BYTE    32,   32,   33,   34,   34,   15,   16,   17

.BYTE   277,  277,  276,  276,  276,  275,  275,  275
.BYTE   275,  274,  274,  274,  253,  253,  253,  253
.BYTE   252,  252,  252,  252,  251,  251,  231,  231
.BYTE   231,  230,  230,  230,  230,  210,  207,  207
.BYTE   207,  207,  167,  167,  167,  167,  147,  147
.BYTE   147,  147,  127,  127,  127,  127,  107,  107
.BYTE   107,  110,   70,   70,   71,   71,   51,   52
.BYTE    52,   53,   53,   34,   35,   35,   36,   37

.BYTE   277,  277,  276,  276,  276,  276,  275,  255
.BYTE   255,  254,  254,  254,  254,  253,  253,  253
.BYTE   253,  252,  252,  252,  232,  232,  231,  231
.BYTE   231,  231,  231,  230,  210,  210,  210,  210
.BYTE   210,  170,  170,  167,  167,  167,  147,  147
.BYTE   147,  147,  147,  127,  130,  130,  130,  110
.BYTE   110,  110,  111,   71,   71,   72,   72,   72
.BYTE    53,   53,   54,   54,   55,   56,   56,   57

.BYTE   257,  257,  256,  256,  256,  256,  255,  255
.BYTE   255,  255,  254,  254,  254,  254,  253,  253
.BYTE   253,  253,  233,  232,  232,  232,  232,  232
.BYTE   231,  231,  211,  211,  211,  211,  211,  210
.BYTE   210,  170,  170,  170,  170,  170,  150,  150
.BYTE   150,  150,  150,  130,  130,  130,  131,  111
.BYTE   111,  111,  111,  112,   72,   72,   73,   73
.BYTE    73,   74,   74,   55,   55,   56,   56,   57

.BYTE   257,  257,  256,  256,  256,  256,  255,  255
.BYTE   255,  255,  255,  254,  254,  254,  254,  253
.BYTE   233,  233,  233,  233,  232,  232,  232,  232
.BYTE   232,  212,  212,  211,  211,  211,  211,  211
.BYTE   171,  171,  171,  171,  171,  171,  151,  151
.BYTE   151,  151,  151,  131,  131,  131,  131,  131
.BYTE   112,  112,  112,  112,  112,  113,   73,   73
.BYTE    74,   74,   75,   75,   75,   76,   76,   77

.BYTE   257,  257,  256,  256,  256,  256,  256,  255
.BYTE   255,  255,  255,  255,  254,  234,  234,  234
.BYTE   234,  233,  233,  233,  233,  233,  232,  232
.BYTE   212,  212,  212,  212,  212,  212,  211,  211
.BYTE   171,  171,  171,  171,  171,  171,  151,  151
```

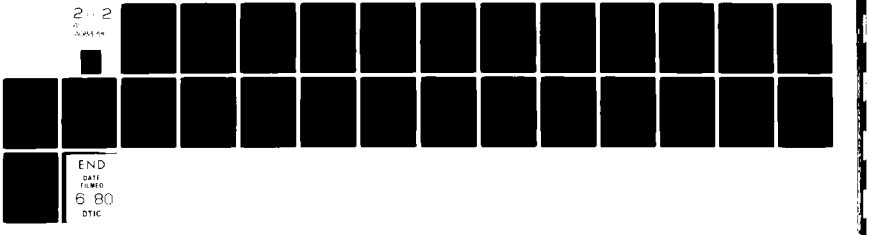
```
.BYTE   151,  151,  151,  151,  131,  132,  132,  132
.BYTE   132,  132,  112,  113,  113,  113,  114,  114
.BYTE   114,   75,   75,   75,   76,   76,   76,   77

.BYTE   257,  257,  256,  256,  256,  256,  256,  255
.BYTE   255,  235,  235,  235,  234,  234,  234,  234
.BYTE   234,  234,  233,  233,  233,  233,  233,  213
.BYTE   213,  212,  212,  212,  212,  212,  212,  172
.BYTE   172,  172,  172,  172,  172,  172,  152,  152
.BYTE   152,  152,  152,  152,  132,  132,  132,  132
.BYTE   133,  133,  133,  113,  113,  114,  114,  114
.BYTE   114,  115,  115,  115,  116,  116,  116,  117

.BYTE   237,  237,  237,  236,  236,  236,  236,  236
.BYTE   235,  235,  235,  235,  235,  234,  234,  234
.BYTE   234,  234,  234,  234,  233,  213,  213,  213
.BYTE   213,  213,  213,  213,  212,  212,  212,  172
.BYTE   172,  172,  172,  172,  172,  172,  152,  152
.BYTE   152,  152,  152,  152,  152,  133,  133,  133
.BYTE   133,  133,  133,  134,  134,  114,  114,  114
.BYTE   115,  115,  115,  116,  116,  116,  117,  117

.BYTE   237,  237,  237,  236,  236,  236,  236,  236
.BYTE   235,  235,  235,  235,  235,  235,  235,  234
.BYTE   234,  234,  234,  234,  214,  214,  213,  213
.BYTE   213,  213,  213,  213,  213,  213,  173,  173
.BYTE   173,  173,  173,  172,  172,  172,  153,  153
.BYTE   153,  153,  153,  153,  153,  153,  133,  133
.BYTE   133,  134,  134,  134,  134,  134,  135,  115
.BYTE   115,  115,  115,  116,  116,  116,  117,  117

.BYTE   237,  237,  237,  236,  236,  236,  236,  236
.BYTE   236,  235,  235,  235,  235,  235,  235,  235
.BYTE   234,  234,  214,  214,  214,  214,  214,  214
.BYTE   213,  213,  213,  213,  213,  173,  173,  173
.BYTE   173,  173,  173,  173,  173,  173,  153,  153
.BYTE   153,  153,  153,  153,  153,  153,  153,  134
.BYTE   134,  134,  134,  134,  134,  135,  135,  135
.BYTE   135,  135,  136,  116,  116,  116,  117,  117

.BYTE   237,  237,  237,  236,  236,  236,  236,  236
.BYTE   236,  236,  235,  235,  235,  235,  235,  235
.BYTE   215,  214,  214,  214,  214,  214,  214,  214
.BYTE   214,  214,  214,  214,  213,  173,  173,  173
.BYTE   173,  173,  173,  173,  173,  173,  173,  153
.BYTE   153,  153,  153,  154,  154,  154,  154,  154
.BYTE   134,  134,  134,  134,  135,  135,  135,  135
.BYTE   135,  136,  136,  136,  136,  136,  137,  137
```

```
.BYTE   237, 237, 237, 236, 236, 236, 236, 236
.BYTE   236, 236, 236, 235, 235, 215, 215, 215
.BYTE   215, 215, 215, 214, 214, 214, 214, 214
.BYTE   214, 214, 214, 214, 174, 174, 174, 174
.BYTE   174, 174, 174, 174, 174, 174, 174, 154
.BYTE   154, 154, 154, 154, 154, 154, 154, 154
.BYTE   154, 154, 135, 135, 135, 135, 135, 135
.BYTE   136, 136, 136, 136, 136, 136, 137, 137

.BYTE   237, 237, 237, 237, 236, 236, 236, 236
.BYTE   236, 236, 216, 216, 215, 215, 215, 215
.BYTE   215, 215, 215, 215, 215, 214, 214, 214
.BYTE   214, 214, 214, 214, 174, 174, 174, 174
.BYTE   174, 174, 174, 174, 174, 174, 174, 154
.BYTE   154, 154, 154, 154, 154, 154, 154, 154
.BYTE   155, 155, 155, 135, 135, 135, 135, 136
.BYTE   136, 136, 136, 136, 136, 137, 137, 137

.BYTE   237, 237, 237, 237, 236, 216, 216, 216
.BYTE   216, 216, 216, 216, 216, 215, 215, 215
.BYTE   215, 215, 215, 215, 215, 215, 215, 215
.BYTE   214, 214, 214, 174, 174, 174, 174, 174
.BYTE   174, 174, 174, 174, 174, 174, 174, 174
.BYTE   154, 154, 154, 154, 154, 155, 155, 155
.BYTE   155, 155, 155, 155, 155, 155, 136, 136
.BYTE   136, 136, 136, 136, 136, 137, 137, 137

.BYTE   217, 217, 217, 217, 216, 216, 216, 216
.BYTE   216, 216, 216, 216, 216, 216, 215, 215
.BYTE   215, 215, 215, 215, 215, 215, 215, 215
.BYTE   215, 215, 175, 174, 174, 174, 174, 174
.BYTE   174, 174, 174, 174, 174, 174, 174, 174
.BYTE   154, 154, 155, 155, 155, 155, 155, 155
.BYTE   155, 155, 155, 155, 155, 156, 156, 156
.BYTE   156, 136, 136, 136, 136, 137, 137, 137

.BYTE   217, 217, 217, 217, 217, 216, 216, 216
.BYTE   216, 216, 216, 216, 216, 216, 216, 215
.BYTE   215, 215, 215, 215, 215, 215, 215, 215
.BYTE   215, 175, 175, 175, 175, 175, 175, 175
.BYTE   175, 175, 175, 175, 175, 175, 175, 175
.BYTE   175, 155, 155, 155, 155, 155, 155, 155
.BYTE   155, 155, 155, 155, 156, 156, 156, 156
.BYTE   156, 156, 156, 156, 157, 157, 157, 157

.BYTE   217, 217, 217, 217, 217, 216, 216, 216
.BYTE   216, 216, 216, 216, 216, 216, 216, 216
```

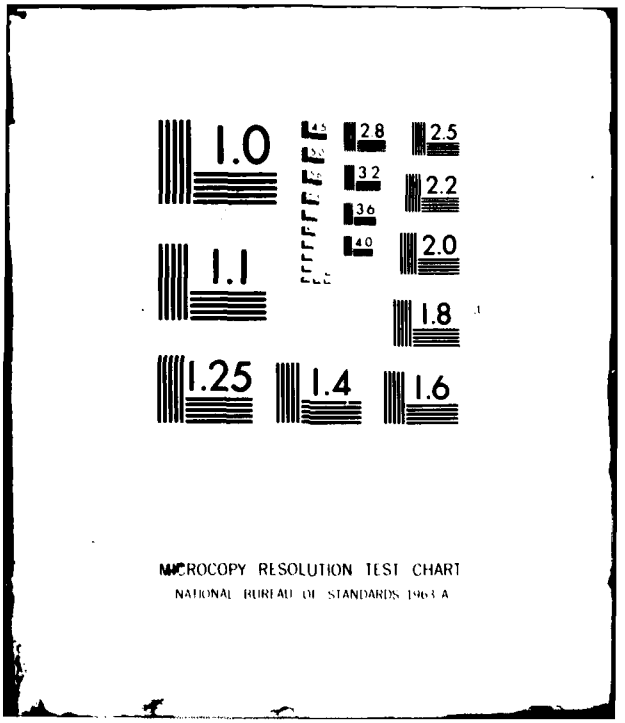
```
.BYTE   215, 215, 215, 215, 215, 215, 215, 215
.BYTE   215, 175, 175, 175, 175, 175, 175, 175
.BYTE   175, 175, 175, 175, 175, 175, 175, 175
.BYTE   175, 155, 155, 155, 155, 155, 155, 155
.BYTE   155, 155, 155, 156, 156, 156, 156, 156
.BYTE   156, 156, 156, 156, 157, 157, 157, 157

.BYTE   217, 217, 217, 217, 217, 217, 216, 216
.BYTE   216, 216, 216, 216, 216, 216, 216, 216
.BYTE   216, 216, 215, 215, 215, 215, 215, 215
.BYTE   175, 175, 175, 175, 175, 175, 175, 175
.BYTE   175, 175, 175, 175, 175, 175, 175, 175
.BYTE   175, 175, 155, 155, 155, 155, 155, 155
.BYTE   155, 156, 156, 156, 156, 156, 156, 156
.BYTE   156, 156, 156, 157, 157, 157, 157, 157

.BYTE   217, 217, 217, 217, 217, 217, 216, 216
.BYTE   216, 216, 216, 216, 216, 216, 216, 216
.BYTE   216, 216, 216, 216, 215, 215, 215, 175
.BYTE   175, 175, 175, 175, 175, 175, 175, 175
.BYTE   175, 175, 175, 175, 175, 175, 175, 175
.BYTE   175, 175, 175, 155, 155, 155, 155, 156
.BYTE   156, 156, 156, 156, 156, 156, 156, 156
.BYTE   156, 156, 156, 157, 157, 157, 157, 157

.BYTE   217, 217, 217, 217, 217, 217, 216, 216
.BYTE   216, 216, 216, 216, 216, 216, 216, 216
.BYTE   216, 216, 216, 216, 216, 216, 176, 175
.BYTE   175, 175, 175, 175, 175, 175, 175, 175
.BYTE   175, 175, 175, 175, 175, 175, 175, 175
.BYTE   175, 175, 175, 155, 156, 156, 156, 156
.BYTE   156, 156, 156, 156, 156, 156, 156, 156
.BYTE   156, 156, 156, 157, 157, 157, 157, 157

.BYTE   217, 217, 217, 217, 217, 217, 217, 216
.BYTE   216, 216, 216, 216, 216, 216, 216, 216
.BYTE   216, 216, 216, 216, 176, 176, 176, 176
.BYTE   176, 176, 176, 175, 175, 175, 175, 175
.BYTE   175, 175, 175, 175, 175, 175, 175, 175
.BYTE   176, 176, 176, 176, 156, 156, 156, 156
.BYTE   156, 156, 156, 156, 156, 156, 156, 156
.BYTE   156, 156, 157, 157, 157, 157, 157, 157

.BYTE   217, 217, 217, 217, 217, 217, 217, 217
.BYTE   216, 216, 216, 216, 216, 216, 216, 216
.BYTE   216, 216, 216, 176, 176, 176, 176, 176
.BYTE   176, 176, 176, 176, 176, 176, 176, 176
.BYTE   176, 176, 176, 176, 176, 176, 176, 176
.BYTE   176, 176, 176, 176, 176, 176, 156, 156
.BYTE   156, 156, 156, 156, 156, 156, 156, 156
.BYTE   156, 157, 157, 157, 157, 157, 157, 157

        .END
```

```
        .TITLE EPROM2 9-SEP-77 COKER
        .GLOBL  EPROM2

EPROM2:
.BYTE   377,  377,  377,  377,  377,  377,  377,  377
.BYTE   377,  377,  377,  377,  377,  377,  377,  377
.BYTE   377,  377,  377,  377,  377,  377,  377,  377
.BYTE   377,  377,  377,  377,  377,  377,  357,  357
.BYTE   357,  357,  357,  357,  357,  357,  357,  357
.BYTE   357,  357,  357,  357,  357,  357,  357,  357
.BYTE   357,  357,  357,  357,  357,  357,  257,  257
.BYTE   257,  257,  237,  237,  237,  237,  137,  117

.BYTE   377,  377,  377,  377,  377,  377,  377,  377
.BYTE   377,  377,  377,  377,  377,  377,  377,  377
.BYTE   377,  377,  377,  377,  377,  377,  377,  377
.BYTE   377,  377,  377,  377,  377,  377,  377,  377
.BYTE   357,  357,  357,  357,  357,  357,  357,  357
.BYTE   357,  357,  357,  357,  357,  357,  357,  357
.BYTE   357,  357,  357,  357,  357,  257,  257,  257
.BYTE   257,  257,  257,  257,  257,  157,  157,  177

.BYTE   377,  377,  377,  377,  377,  377,  377,  377
.BYTE   377,  377,  377,  377,  377,  377,  377,  377
.BYTE   377,  377,  377,  377,  377,  377,  377,  377
.BYTE   377,  377,  377,  377,  377,  377,  377,  377
.BYTE   377,  377,  357,  357,  357,  357,  357,  357
.BYTE   357,  357,  357,  357,  357,  357,  357,  357
.BYTE   357,  357,  357,  357,  357,  257,  257,  257
.BYTE   257,  257,  257,  257,  257,  277,  277,  177

.BYTE   377,  377,  377,  377,  377,  377,  377,  377
.BYTE   377,  377,  377,  377,  377,  377,  377,  377
.BYTE   377,  377,  377,  377,  377,  377,  377,  377
.BYTE   377,  377,  377,  377,  377,  377,  377,  377
.BYTE   377,  377,  377,  377,  357,  357,  357,  357
.BYTE   357,  357,  357,  357,  357,  357,  357,  357
.BYTE   357,  357,  357,  357,  357,  257,  257,  257
.BYTE   257,  257,  257,  277,  277,  277,  277,  277

.BYTE   377,  377,  377,  377,  377,  377,  377,  377
.BYTE   377,  377,  377,  377,  377,  377,  377,  377
.BYTE   377,  377,  377,  377,  377,  377,  377,  377
.BYTE   377,  377,  377,  377,  377,  377,  377,  377
.BYTE   377,  377,  377,  377,  377,  377,  377,  377
.BYTE   357,  357,  357,  357,  357,  357,  357,  357
.BYTE   357,  357,  357,  357,  357,  257,  257,  277
.BYTE   277,  277,  277,  277,  277,  277,  277,  277
```

AD-A083 991 AIR FORCE AVIONICS LAB WRIGHT-PATTERSON AFB OH F/G 9/2
LOG POLAR QUANTIZATION.(U)
OCT 79 R T COKER, G D COUTURIER
UNCLASSIFIED AFAL-TR-79-1075 NL



END
DATE FILMED
6 80
DTIC


1.0
1.1
1.25
2.8
3.2
3.6
4.0
2.5
2.2
2.0
1.8
1.4
1.6
MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS 1963 A

```
.BYTE   377,  377,  377,  377,  377,  377,  377,  377
.BYTE   377,  377,  377,  377,  377,  377,  377,  377
.BYTE   377,  377,  377,  377,  377,  377,  377,  377
.BYTE   377,  377,  377,  377,  377,  377,  377,  377
.BYTE   377,  377,  377,  377,  377,  377,  377,  377
.BYTE   377,  377,  377,  377,  377,  377,  377,  377
.BYTE   377,  377,  377,  377,  377,  377,  277,  277
.BYTE   277,  277,  277,  277,  277,  277,  277,  277

.BYTE   377,  377,  377,  377,  377,  377,  377,  377
.BYTE   377,  377,  377,  377,  377,  377,  377,  377
.BYTE   377,  377,  377,  377,  377,  377,  377,  377
.BYTE   377,  377,  377,  377,  377,  377,  377,  377
.BYTE   377,  377,  377,  377,  377,  377,  377,  377
.BYTE   377,  377,  377,  377,  377,  377,  377,  377
.BYTE   377,  377,  377,  377,  377,  377,  377,  277
.BYTE   277,  277,  277,  277,  277,  277,  277,  277

.BYTE   377,  377,  377,  377,  377,  377,  377,  377
.BYTE   377,  377,  377,  377,  377,  377,  377,  377
.BYTE   377,  377,  377,  377,  377,  377,  377,  377
.BYTE   377,  377,  377,  377,  377,  377,  377,  377
.BYTE   377,  377,  377,  377,  377,  377,  377,  377
.BYTE   377,  377,  377,  377,  377,  377,  377,  377
.BYTE   377,  377,  377,  377,  377,  377,  377,  377
.BYTE   277,  277,  277,  277,  277,  277,  277,  277

.BYTE   377,  377,  377,  377,  377,  377,  377,  377
.BYTE   377,  377,  377,  377,  377,  377,  377,  377
.BYTE   377,  377,  377,  377,  377,  377,  377,  377
.BYTE   377,  377,  377,  377,  377,  377,  377,  377
.BYTE   377,  377,  377,  377,  377,  377,  377,  377
.BYTE   377,  377,  377,  377,  377,  377,  377,  377
.BYTE   377,  377,  377,  377,  377,  377,  377,  377
.BYTE   377,  377,  277,  277,  277,  277,  277,  277

.BYTE   377,  377,  377,  377,  377,  377,  377,  377
.BYTE   377,  377,  377,  377,  377,  377,  377,  377
.BYTE   377,  377,  377,  377,  377,  377,  377,  377
.BYTE   377,  377,  377,  377,  377,  377,  377,  377
.BYTE   377,  377,  377,  377,  377,  377,  377,  377
.BYTE   377,  377,  377,  377,  377,  377,  377,  377
.BYTE   377,  377,  377,  377,  377,  377,  377,  377
.BYTE   377,  377,  377,  377,  377,  377,  377,  277

.BYTE   377,  377,  377,  377,  377,  377,  377,  377
.BYTE   377,  377,  377,  377,  377,  377,  377,  377
```

```
.BYTE   377,  377,  377,  377,  377,  377,  377,  377
.BYTE   377,  377,  377,  377,  377,  377,  377,  377
.BYTE   377,  377,  377,  377,  377,  377,  377,  377
.BYTE   377,  377,  377,  377,  377,  377,  377,  377
.BYTE   377,  377,  377,  377,  377,  377,  377,  377
.BYTE   377,  377,  377,  377,  377,  377,  377,  377

.BYTE   377,  377,  377,  377,  377,  377,  377,  377
.BYTE   377,  377,  377,  377,  377,  377,  377,  377
.BYTE   377,  377,  377,  377,  377,  377,  377,  377
.BYTE   377,  377,  377,  377,  377,  377,  377,  377
.BYTE   377,  377,  377,  377,  377,  377,  377,  377
.BYTE   377,  377,  377,  377,  377,  377,  377,  377
.BYTE   377,  377,  377,  377,  377,  377,  377,  377
.BYTE   377,  377,  377,  377,  377,  377,  377,  377

.BYTE   377,  377,  377,  377,  377,  377,  377,  377
.BYTE   377,  377,  377,  377,  377,  377,  377,  377
.BYTE   377,  377,  377,  377,  377,  377,  377,  377
.BYTE   377,  377,  377,  377,  377,  377,  377,  377
.BYTE   377,  377,  377,  377,  377,  377,  377,  377
.BYTE   377,  377,  377,  377,  377,  377,  377,  377
.BYTE   377,  377,  377,  377,  377,  377,  377,  377
.BYTE   377,  377,  377,  377,  377,  377,  377,  377

.BYTE   377,  377,  377,  377,  377,  377,  377,  377
.BYTE   377,  377,  377,  377,  377,  377,  377,  377
.BYTE   377,  377,  377,  377,  377,  377,  377,  377
.BYTE   377,  377,  377,  377,  377,  377,  377,  377
.BYTE   377,  377,  377,  377,  377,  377,  377,  377
.BYTE   377,  377,  377,  377,  377,  377,  377,  377
.BYTE   377,  377,  377,  377,  377,  377,  377,  377
.BYTE   377,  377,  377,  377,  377,  377,  377,  377

.BYTE   377,  377,  377,  377,  377,  377,  377,  377
.BYTE   377,  377,  377,  377,  377,  377,  377,  377
.BYTE   377,  377,  377,  377,  377,  377,  377,  377
.BYTE   377,  377,  377,  377,  377,  377,  377,  377
.BYTE   377,  377,  377,  377,  377,  377,  377,  377
.BYTE   377,  377,  377,  377,  377,  377,  377,  377
.BYTE   377,  377,  377,  377,  377,  377,  377,  377
.BYTE   377,  377,  377,  377,  377,  377,  377,  377

.BYTE   377,  377,  377,  377,  377,  377,  377,  377
.BYTE   377,  377,  377,  377,  377,  377,  377,  377
.BYTE   377,  377,  377,  377,  377,  377,  377,  377
.BYTE   377,  377,  377,  377,  377,  377,  377,  377
.BYTE   377,  377,  377,  377,  377,  377,  377,  377
```

```
.BYTE   377, 377, 377, 377, 377, 377, 377, 377
.BYTE   377, 377, 377, 377, 377, 377, 377, 377
.BYTE   377, 377, 377, 377, 377, 377, 377, 377

.BYTE   371, 371, 371, 371, 371, 371, 371, 371
.BYTE   370, 370, 370, 370, 370, 370, 370, 370
.BYTE   370, 366, 366, 366, 366, 366, 366, 366
.BYTE   366, 366, 366, 366, 366, 366, 366, 366
.BYTE   366, 366, 366, 366, 366, 366, 366, 366
.BYTE   366, 366, 366, 366, 366, 366, 366, 364
.BYTE   364, 364, 364, 364, 364, 364, 364, 364
.BYTE   364, 365, 365, 365, 365, 365, 365, 365

.BYTE   371, 371, 371, 371, 371, 371, 371, 371
.BYTE   371, 370, 370, 370, 370, 370, 370, 366
.BYTE   366, 366, 366, 366, 366, 366, 366, 366
.BYTE   366, 366, 366, 366, 366, 366, 366, 366
.BYTE   366, 366, 366, 366, 366, 366, 366, 366
.BYTE   366, 366, 366, 366, 366, 366, 366, 366
.BYTE   364, 364, 364, 364, 364, 364, 364, 364
.BYTE   365, 365, 365, 365, 365, 365, 365, 365

.BYTE   371, 371, 371, 371, 371, 371, 371, 371
.BYTE   371, 371, 370, 370, 370, 366, 366, 366
.BYTE   366, 366, 366, 366, 366, 366, 366, 366
.BYTE   366, 366, 366, 366, 366, 366, 366, 366
.BYTE   366, 366, 366, 366, 366, 366, 366, 366
.BYTE   366, 366, 366, 366, 366, 366, 366, 366
.BYTE   366, 366, 364, 364, 364, 364, 364, 365
.BYTE   365, 365, 365, 365, 365, 365, 365, 365

.BYTE   371, 371, 371, 371, 371, 371, 371, 371
.BYTE   371, 371, 367, 366, 366, 366, 366, 366
.BYTE   366, 366, 366, 366, 366, 366, 366, 366
.BYTE   366, 366, 366, 366, 366, 366, 366, 366
.BYTE   366, 366, 366, 366, 366, 366, 366, 366
.BYTE   366, 366, 366, 366, 366, 366, 366, 366
.BYTE   366, 366, 366, 366, 366, 364, 365, 365
.BYTE   365, 365, 365, 365, 365, 365, 365, 365

.BYTE   371, 371, 371, 371, 371, 371, 367, 367
.BYTE   367, 367, 367, 367, 366, 366, 366, 366
.BYTE   366, 366, 366, 366, 366, 366, 366, 366
.BYTE   366, 366, 366, 366, 366, 366, 366, 366
.BYTE   366, 366, 366, 366, 366, 366, 366, 366
.BYTE   366, 366, 366, 366, 366, 366, 366, 366
.BYTE   366, 366, 366, 366, 366, 367, 367, 367
.BYTE   365, 365, 365, 365, 365, 365, 365, 365
```

```
.BYTE   367, 367, 367, 367, 367, 367, 367, 367
.BYTE   367, 367, 367, 367, 367, 366, 366, 366
.BYTE   366, 366, 366, 366, 366, 366, 366, 366
.BYTE   366, 366, 366, 366, 366, 366, 366, 366
.BYTE   366, 366, 366, 366, 366, 366, 366, 366
.BYTE   366, 366, 366, 366, 366, 366, 366, 366
.BYTE   366, 366, 366, 366, 367, 367, 367, 367
.BYTE   367, 367, 367, 367, 367, 367, 367, 367

.BYTE   367, 367, 367, 367, 367, 367, 367, 367
.BYTE   367, 367, 367, 367, 367, 367, 366, 366
.BYTE   366, 366, 366, 366, 366, 366, 366, 366
.BYTE   366, 366, 366, 366, 366, 366, 366, 366
.BYTE   366, 366, 366, 366, 366, 366, 366, 366
.BYTE   366, 366, 366, 366, 366, 366, 366, 366
.BYTE   366, 366, 366, 367, 367, 367, 367, 367
.BYTE   367, 367, 367, 367, 367, 367, 367, 367

.BYTE   367, 367, 367, 367, 367, 367, 367, 367
.BYTE   367, 367, 367, 367, 367, 367, 367, 367
.BYTE   366, 366, 366, 366, 366, 366, 366, 366
.BYTE   366, 366, 366, 366, 366, 366, 366, 366
.BYTE   366, 366, 366, 366, 366, 366, 366, 366
.BYTE   366, 366, 366, 366, 366, 366, 366, 366
.BYTE   366, 367, 367, 367, 367, 367, 367, 367
.BYTE   367, 367, 367, 367, 367, 367, 367, 367

.BYTE   367, 367, 367, 367, 367, 367, 367, 367
.BYTE   367, 367, 367, 367, 367, 367, 367, 367
.BYTE   367, 367, 366, 366, 366, 366, 366, 366
.BYTE   366, 366, 366, 366, 366, 366, 366, 366
.BYTE   366, 366, 366, 366, 366, 366, 366, 366
.BYTE   366, 366, 366, 366, 366, 366, 366, 367
.BYTE   367, 367, 367, 367, 367, 367, 367, 367
.BYTE   367, 367, 367, 367, 367, 367, 367, 367

.BYTE   367, 367, 367, 367, 367, 367, 367, 367
.BYTE   367, 367, 367, 367, 367, 367, 367, 367
.BYTE   367, 367, 367, 367, 366, 366, 366, 366
.BYTE   366, 366, 366, 366, 366, 366, 366, 366
.BYTE   366, 366, 366, 366, 366, 366, 366, 366
.BYTE   366, 366, 366, 366, 366, 367, 367, 367
.BYTE   367, 367, 367, 367, 367, 367, 367, 367
.BYTE   367, 367, 367, 367, 367, 367, 367, 367

.BYTE   367, 367, 367, 367, 367, 367, 367, 367
.BYTE   367, 367, 367, 367, 367, 367, 367, 367
```

```
.BYTE   367,  367,  367,  367,  367,  367,  367,  366
.BYTE   366,  366,  366,  366,  366,  366,  366,  366
.BYTE   366,  366,  366,  366,  366,  366,  366,  366
.BYTE   366,  366,  367,  367,  367,  367,  367,  367
.BYTE   367,  367,  367,  367,  367,  367,  367,  367
.BYTE   367,  367,  367,  367,  367,  367,  367,  367

.BYTE   367,  367,  367,  367,  367,  367,  367,  367
.BYTE   367,  367,  367,  367,  367,  367,  367,  367
.BYTE   367,  367,  367,  367,  367,  367,  367,  367
.BYTE   367,  367,  367,  367,  366,  366,  366,  366
.BYTE   366,  366,  366,  366,  366,  367,  367,  367
.BYTE   367,  367,  367,  367,  367,  367,  367,  367
.BYTE   367,  367,  367,  367,  367,  367,  367,  367
.BYTE   367,  367,  367,  367,  367,  367,  367,  367

.BYTE   367,  367,  367,  367,  367,  367,  367,  367
.BYTE   367,  367,  367,  367,  367,  367,  367,  367
.BYTE   367,  367,  367,  367,  367,  367,  367,  367
.BYTE   367,  367,  367,  367,  367,  367,  367,  367
.BYTE   367,  367,  367,  367,  367,  367,  367,  367
.BYTE   367,  367,  367,  367,  367,  367,  367,  367
.BYTE   367,  367,  367,  367,  367,  367,  367,  367
.BYTE   367,  367,  367,  367,  367,  367,  367,  367

.BYTE   367,  367,  367,  367,  367,  367,  367,  367
.BYTE   367,  367,  367,  367,  367,  367,  367,  367
.BYTE   367,  367,  367,  367,  367,  367,  367,  367
.BYTE   367,  367,  367,  367,  367,  367,  367,  367
.BYTE   367,  367,  367,  367,  367,  367,  367,  367
.BYTE   367,  367,  367,  367,  367,  367,  367,  367
.BYTE   367,  367,  367,  367,  367,  367,  367,  367
.BYTE   367,  367,  367,  367,  367,  367,  367,  367

.BYTE   367,  367,  367,  367,  367,  367,  367,  367
.BYTE   367,  367,  367,  367,  367,  367,  367,  367
.BYTE   367,  367,  367,  367,  367,  367,  367,  367
.BYTE   367,  367,  367,  367,  367,  367,  367,  367
.BYTE   367,  367,  367,  367,  367,  367,  367,  367
.BYTE   367,  367,  367,  367,  367,  367,  367,  367
.BYTE   367,  367,  367,  367,  367,  367,  367,  367
.BYTE   367,  367,  367,  367,  367,  367,  367,  367

.BYTE   367,  367,  367,  367,  367,  367,  367,  367
.BYTE   367,  367,  367,  367,  367,  367,  367,  367
.BYTE   367,  367,  367,  367,  367,  367,  367,  367
.BYTE   367,  367,  367,  367,  367,  367,  367,  367
.BYTE   367,  367,  367,  367,  367,  367,  367,  367
.BYTE   367,  367,  367,  367,  367,  367,  367,  367
.BYTE   367,  367,  367,  367,  367,  367,  367,  367
.BYTE   367,  367,  367,  367,  367,  367,  367,  367

        .END
```

```
        .TITLE ISWAB  IO=ISWAB(II) 24-JUN-77 COKER
        .MCALL  ..V2..  .REGDEF
        ..V2..
        .REGDEF
        .GLOBL  ISWAB

ISWAB:  MOV     (R5)+,  R0      ;GET # OF ARGUMENTS
        MOV     @(R5),  R0      ;GET ARG
        SWAB    R0              ;SWAP BYTES
        RTS     PC              ;RETURN
        .END
```

```
        .TITLE  CLEAR 8TH BIT 15-DEC-76 COKER
;FORTRTAN FUNCTION ROUTINE 'I8CLR' IN MACRO
;FORTRAN EX: I=I8CLR(I) - WHERE THE 8TH BIT IS CLEARED

        .MCALL  ..V2..  .REGDEF
        ..V2..
        .REGDEF

        .GLOBL  I8CLR
I8CLR:  MOV     (R5)+   ,R0     ;GET # OF ARGUEMENTS
        CMPB    #1      ,R0     ;1 ARG ?
        BNE     ERR             ;IF NOT THEN ERROR
        MOV     @(R5)+  ,R0     ;GET 'I'
        BIC     #200    ,R0     ;CLEAR BIT # 8
        RTS     PC              ;RETURN

ERR:    JMP     @#4             ;ABORT TO MONITOR
        .END
```

```
        .TITLE  BIT REVERSAL ROUTINE 14 DEC76 COKER
;FORTRAN FUNCTION ROUTINE 'IBITR' IN MACRO
;FORTRAN EX: IBR=IBITR(I,NR) - WHERE:
;'IBR' IS THE BIT REVERSED VERSION OF 'I' &
;'NB' IS THE NUMBER OF BITS REVERSED

        .MCALL  ..V2..  .REGDEF
        ..V2..
        .REGDEF
        .GLOBL  IBITR

IBITR:  MOV     (R5)+   ,R0     ;GET # OF ARGUMENTS
        CMPB    #2      ,R0     ;2 ARG ?
        BNE     ERR             ;IF NOT THEN ERROR
        MOV     @(R5)+  ,R1     ;GET 'I'
        MOV     @(R5)+  ,R2     ;GET 'NB'

        CLR     R0              ;CLEAR 'IBR'

LOOP1:  ROR     R1              ;'I' BIT OUT
        ROL     R0              ;'IBR' BIT IN
        DEC     R2              ;DEC BIT COUNT
        BEQ     DONE            ;JUMP IF COUNT ZERO
        JMP     LOOP1           ;LOOP IF NOT ZERO

DONE:                           ;RESULTS IN R0 ?
        RTS     PC              ;RETURN

ERR:    JMP     @#4             ;ABORT TO MONITOR
        .END
```

```
**** LOGPOL MAIN INTERCONNECT PROGRAM ****
AND Y1          Y9                                                  TD00
OR  Y2          TA13                                                TD01
OR  Y3          TA13                                                TD02
OR  Y4          TA13                                                TD03
OR  Y13         TA12                                                TD04
OR  Y14         TA12                                                TD05
AND Y5          Y10                                                 TD08
AND Y6          Y11                                                 TD09
AND Y11         Y7                                                  TD10
AND Y12         Y8                                                  TD11
OR  Y15         TA12                                                TD12
OR  Y16         TA12                                                TD13
A381BVIA0       AVIA0       BVIA1       AVIA1       BVIA2           ADD1        ADD2
CONTAVIA2       BVIA3       AVIA3       LF          LFNOT           ADD3        ADD4
CONTBVIA0N      AVIA0N      BVIA1N      AVIA1N      BVIA2N          CA381
CONTAVIA2N      BVIA3N      AVIA3N
B381BVIA4       AVIA4       BVIA5       AVIA5       BVIA6           ADD5        ADD6
CONTAVIA6       LF          LFNOT       BVIA4N      AVIA4N          ADD7
CONTBVIA5N      AVIA5N      BVIA6N      AVIA6N      CA381
A1X7ADD1        ADD2        ADD3        ADD4        ADD5            TA08        TA09
CONTADD6        ADD7        C0                                      SUM3        SUM4
CONT                                                                SUM5        SUM6
CONT                                                                SUM7
OR  SUM3        OR1                                                 TA10
OR  SUM4        OR1                                                 TA11
AND SUM5        OR1N                                                TA12
INV OR1                                                             OR1N
OR  SUM6        OR1                                                 TA13
OR  SUM7        AND1                                                OR1
AND SUM5        SUM6                                                AND1
MA85AVIA0       AVIA1       AVIA2       AVIA3       AVIA0N          AGTB        ALTB
CONTAVIA1N      AVIA2N      AVIA3N      BVIA0       BVIA1           AEQB
CONTBVIA2       BVIA3       BVIA0N      BVIA1N      BVIA2N
CONTBVIA3N
MB85AVIA4       AVIA5       AVIA6       AVIA4N      AVIA5N          LF          LFNOT
CONTAVIA6N      BVIA4       BVIA5       BVIA6       BVIA4N
CONTBVIA5N      BVIA6N      AGTB        ALTB        AEQB
INV AVIA0                                                           AVIA0N
```

```
INV AVIA1                                                    AVIA1N
INV AVIA2                                                    AVIA2N
INV AVIA3                                                    AVIA3N
INV AVIA4                                                    AVIA4N
INV AVIA5                                                    AVIA5N
INV AVIA6                                                    AVIA6N
INV BVIA0                                                    BVIA0N
INV BVIA1                                                    BVIA1N
INV BVIA2                                                    BVIA2N
INV BVIA3                                                    BVIA3N
INV BVIA4                                                    BVIA4N
INV BVIA5                                                    BVIA5N
INV BVIA6                                                    BVIA6N
INV AVIP0                                                    AVIP0N
INV AVIP1                                                    AVIP1N
INV AVIP2                                                    AVIP2N
INV AVIP3                                                    AVIP3N
INV AVIP4                                                    AVIP4N
INV AVIP5                                                    AVIP5N
INV AVIP6                                                    AVIP6N
INV BVIP0                                                    BVIP0N
INV BVIP1                                                    BVIP1N
INV BVIP2                                                    BVIP2N
INV BVIP3                                                    BVIP3N
INV BVIP4                                                    BVIP4N
INV BVIP5                                                    BVIP5N
INV BVIP6                                                    BVIP6N
INV PFFNOT                                                   PFF
C381BVIP0      AVIP0      BVIP1      AVIP1      BVIP2      C0         C1
CONTAVIP2      BVIP3      AVIP3      LF         LFNOT      C2         C3
CONTBVIP0N     AVIP0N     BVIP1N     AVIP1N     BVIP2N     CC381
CONTAVIP2N     BVIP3N     AVIP3N     ADD1
B381BVIP4      AVIP4      BVIP5      AVIP5      BVIP6      C4         C5
CONTAVIP6      LF         LFNOT      BVIP4N     AVIP4N     PFFNOT
CONTBVIP5N     AVIP5N     BVIP6N     AVIP6N     CC381
MH87C0         C1         C2         C3         C4         TA00       TA01
CONTC5         PFFNOT                                       TA02       TA03
CONT                                                        TA04       TA05
A157AVIA0      AVIA1      AVIA2      AVIA3      AVIA4      X1         X2
```

```
CONTAVIA5   AVIA6   BVIA0   BVIA1   BVIA2   X3      X4
CONTBVIA3   BVIA4   BVIA5   BVIA6   LF      X5      X6
CONTLFNOT                                   X7
A157AVIP0   AVIP1   AVIP2   AVIP3   AVIP4   X8      X9
CONTAVIP5   AVIP6   BVIP0   BVIP1   BVIP2   X10     X11
CONTBVIP3   BVIP4   BVIP5   BVIP6   LF      X12     X13
CONTLFNOT                                   X14
A1X6TD00    TD01    TD02    TD03    TD04    ASUM1   ASUM2
CONTTD05                                    ASUM3   ASUM4
CONT                                        ASUM5   ASUM6
CONT                                        ASUM7
LS83X1      X2      X3      X4      X5      V08     V09
CONTX6      X7      TD08    TD09    TD10    V10     V11
CONTTD11    TD12    TD13                    V12     V13
CONT                                        V14
E381ASUM1   X8      ASUM2   X9      ASUM3   V00     V01
CONTX10     ASUM4   X11     PFF     PFFNOT  V02     V03
CONT                                        CE381
F381ASUM5   X12     ASUM6   X13     X14     V04     V05
CONTPFF     PFFNOT  CE381   ASUM7           V06
END
```

```
MAC A381
DEFAB0          A0          B1          A1          B2          F0          F1
DEFAA2          B3          A3          S0          S1          F2          F3
DEFAB0N         A0N         B1N         A1N         B2N         CN4
DEFAA2N         B3N         A3N
SB11B0          A0          B0N         A0N         S0          NOR11       F0
CONTS1
SB11B1          A1          B1N         A1N         S0          NOR12       NOR22
CONTS1
SB11B2          A2          B2N         A2N         S0          NOR13       NOR23
CONTS1
SB11B3          A3          B3N         A3N         S0          NOR14       NOR24
CONTS1
SB41NOR11       F0          NOR22                                F1
SB51F0          NOR12       NOR11       NOR22       NOR23       F2
SB61NOR11       NOR12       NOR13       NOR14       F0          F3          CN4
CONTNOR22       NOR23       NOR24
END
```

```
MAC B381
DEFAB0          A0          B1          A1          B2          F0          F1
DEFAA2          S0          S1          B0N         A0N         F2
DEFAB1N         A1N         B2N         A2N         CN
SB12B0          A0          B0N         A0N         S0          NOR11       NOR21
CONTS1
SB12B1          A1          B1N         A1N         S0          NOR12       NOR22
CONTS1
SB12B2          A2          B2N         A2N         S0          NOR13       NOR23
CONTS1
SB32CN          NOR21                                            F0
SB42NOR11       NOR21       NOR22       CN                       F1
SB52NOR21       NOR12       NOR11       NOR22       NOR23       F2
CONTCN
END
```

```
MAC C3B1
DEFAB0      A0      B1      A1      B2      F0      F1
DEFAA2      B3      A3      S0      S1      F2      F3
DEFAB0N     A0N     B1N     A1N     B2N     CN4
DEFAA2N     B3N     A3N     CN
SB13B0      A0      B0N     A0N     S0      NOR11   NOR21
CONTS1
SB13B1      A1      B1N     A1N     S0      NOR12   NOR22
CONTS1
SB13B2      A2      B2N     A2N     S0      NOR13   NOR23
CONTS1
SB13B3      A3      B3N     A3N     S0      NOR14   NOR24
CONTS1                                              F0
SB33CN      NOR21                                   F1
SB43NOR11   NOR21   NOR22   CN                      F2
SB53NOR21   NOR12   NOR11   NOR22   NOR23
CONTCN                                              F3      CN4
SB63NOR11   NOR12   NOR13   NOR14   NOR21
CONTNOR22   NOR23   NOR24   CN
END
```

```
MAC E3B1
DEFAB0      A0      B1      A1      B2      F0      F1
DEFAA2      B3      A3      S0      SONOT   F2      F3
DEFA                                        CN4
SB15B0      A0      S0      SONOT           NOR11   NOR21
SB15B1      A1      S0      SONOT           NOR12   NOR22
SB15B2      A2      S0      SONOT           NOR13   NOR23
SB15B3      A3      S0      SONOT           NOR14   NOR24
SB35S0      NOR21                                   F0
SB45NOR11   NOR21   NOR22   SONOT                   F1
SB55NOR21   NOR12   NOR11   NOR22   NOR23           F2
CONTSONOT                                           F3      CN4
SB65NOR11   NOR12   NOR13   NOR14   NOR21
CONTNOR22   NOR23   NOR24   SONOT   S0
END
```

```
MAC F381
DEFABO          A0          B1          A1          A2          F0          F1
DEFASO          SONOT       CN          B2                      F2
SB16B0          A0          S0          SONOT                   NOR11       NOR21
SB16B1          A1          S0          SONOT                   NOR12       NOR22
SB16B2          A2          S0          SONOT                   NOR13       NOR23
SB36CN          NOR21                                           F0
SB46NOR11       NOR21       NOR22       CN                      F1
SB56NOR21       NOR12       NOR11       NOR22       NOR23       F2
CONTCN
END
```

```
MAC A157
DEFAA1          A2          A3          A4          A5          Y1          Y2
DEFAA6          A7          B1          B2          B3          Y3          Y4
DEFAB4          B5          B6          B7          S           Y5          Y6
DEFASNOT                                                        Y7
AND  A1         SNOT                                            AND1
AND  B1         S                                               AND2
AND  A2         SNOT                                            AND3
AND  B2         S                                               AND4
AND  A3         SNOT                                            AND5
AND  B3         S                                               AND6
AND  A4         SNOT                                            AND7
AND  B4         S                                               AND8
AND  A5         SNOT                                            AND9
AND  B5         S                                               AND10
AND  A6         SNOT                                            AND11
AND  B6         S                                               AND12
AND  A7         SNOT                                            AND13
AND  B7         S                                               AND14
OR   AND1       AND2                                            Y1
OR   AND3       AND4                                            Y2
OR   AND5       AND6                                            Y3
OR   AND7       AND8                                            Y4
OR   AND9       AND10                                           Y5
OR   AND11      AND12                                           Y6
OR   AND13      AND14                                           Y7
END
```

```
MAC LS83
DEFAA1      A2      A3      A4      A5      SUM1    SUM2
DEFAA6      A7      B1      B2      B3      SUM3    SUM4
DEFAB4      B5      B6                      SUM5    SUM6
DEFA                                        SUM7
INV A2                                      A2NOT
INV A4                                      A4NOT
INV A6                                      A6NOT
INV B2                                      B2NOT
INV B4                                      B4NOT
INV B6                                      B6NOT
SB83A1      B1                              O11     O21
CONT                                        O31     C1
SA83A2NOT   B2NOT   C1                      O12     O22
CONT                                        O32     O42
CONT                                        C2
SA83A3      B3      C2                      O13     O23
CONT                                        O33     O43
CONT                                        C3
SA83A4NOT   B4NOT   C3                      O14     O24
CONT                                        O34     O44
CONT                                        C4
SA83A5      B5      C4                      O15     O25
CONT                                        O35     O45
CONT                                        C5
SA83A6NOT   B6NOT   C5                      O16     O26
CONT                                        O36     O46
CONT                                        C6
SC83A7      C6                              SUM7    UFF
NOR O11     O21     O31     C1              NOR1
NOR O12     O22     O32     O42     UFF     SUM2
NOR O13     O23     O33     O43             NOR2
NOR O14     O24     O34     O44     UFF     SUM4
NOR O15     O25     O35     O45             NOR3
NOR O16     O26     O36     O46     UFF     SUM6
NOR NOR1    UFF                             SUM1
NOR NOR2    UFF                             SUM3
NOR NOR3    UFF                             SUM5
END
```

```
MAC A1X6
DEFAA1      A2          A3          A4          A5          SUM1        SUM2
DEFAA6                                                      SUM3        SUM4
DEFA                                                        SUM5        SUM6
DEFA                                                        SUM7
INV A1                                                      SUM1
INV A2                                                      A2N
AND A2N     SUM1                                            AND1
NOR A2N     SUM1        AND1                                C2
NOR AND1    C2                                              SUM2
AND C2      C3                                              AND2
AND A3      C3                                              AND3
NANDA3      C2                                              C3
OR  AND2    AND3                                            SUM3
INV A4                                                      A4N
AND C3      A4N                                             AND5
NOR AND5    A4N         C3                                  C4
NOR AND5    C4                                              SUM4
AND C4      C5                                              AND6
AND A5      C5                                              AND7
NANDA5      C4                                              C5
OR  AND6    AND7                                            SUM5
INV A6                                                      A6N
AND C5      A6N                                             AND9
NOR AND9    A6N         C5                                  C6
NOR AND9    C6                                              SUM6
INV C6                                                      SUM7
END
```

```
MAC A1X7
DEFAA1          A2          A3          A4          A5          SUM1        SUM2
DEFAA6          A7          C                                   SUM3        SUM4
DEFA                                                            SUM5        SUM6
DEFA                                                            SUM7
AND C           C1                                              AND1
AND A1          C1                                              AND2
NANDA1          C                                               C1
OR  AND1        AND2                                            SUM1
INV A2                                                          A2N
AND A2N         C1                                              AND3
NOR AND3        A2N         C1                                  C2
NOR AND3        C2                                              SUM2
AND C2          C3                                              AND4
AND A3          C3                                              AND5
NANDA3          C2                                              C3
OR  AND4        AND5                                            SUM3
INV A4                                                          A4N
AND A4N         C3                                              AND6
NOR AND6        A4N         C3                                  C4
NOR AND6        C4                                              SUM4
AND C4          C5                                              AND7
AND A5          C5                                              AND8
NANDA5          C4                                              C5
OR  AND7        AND8                                            SUM5
INV A6                                                          A6N
AND A6N         C5                                              AND9
NOR AND9        A6N         C5                                  C6
NOR AND9        C6                                              SUM6
AND C6          C7                                              AND10
AND A7          C7                                              AND11
NANDA7          C6                                              C7
OR  AND10       AND11                                           SUM7
END
```

```
MAC MH87
DEFAA1          A2          A3          A4          A5          Y1          Y2
DEFAA6          PFFNOT                                          Y3          Y4
DEFA                                                            Y5          Y6
EOR A1          PFFNOT                                          Y1
EOR A2          PFFNOT                                          Y2
EOR A3          PFFNOT                                          Y3
EOR A4          PFFNOT                                          Y4
EOR A5          PFFNOT                                          Y5
EOR A6          PFFNOT                                          Y6
END
```

```
MAC MA85
DEFAA0      A1      A2      A3      A0N     AGTB    ALTB
DEFAA1N     A2N     A3N     B0      B1      AEQB
DEFAB2      B3      B0N     B1N     B2N
DEFAB3N
AND A3N     B3                                      AND1
AND A3      B3N                                     AND2
AND A2N     B2                                      AND3
AND A2      B2N                                     AND4
AND A1N     B1                                      AND5
AND A1      B1N                                     AND6
AND A0N     B0                                      AND7
AND A0      B0N                                     AND8
NOR AND1    AND2                                    NOR1
NOR AND3    AND4                                    NOR2
NOR AND5    AND6                                    NOR3
NOR AND7    AND8                                    NOR4
AND A2N     B2      NOR1                            AND9
AND A2      B2N     NOR1                            AND10
AND A1N     B1      NOR1    NOR2                    AND11
AND A1      B1N     NOR1    NOR2                    AND12
AND A0N     B0      NOR1    NOR2    NOR3            AND13
AND A0      B0N     NOR1    NOR2    NOR3            AND14
AND NOR1    NOR2    NOR3    NOR4                    AEQB
OR  AND2    AND10   AND12   AND14                   AGTB
OR  AND1    AND9    AND11   AND13                   ALTB
END
```

```
MAC MB85
DEFAA0          A1          A2          AON         A1N         LF          LFNOT
DEFAA2N         B0          B1          B2          BON
DEFAB1N         B2N         AGTB        ALTB        AEQB
AND A2N         B2                                              AND1
AND A2          B2N                                             AND2
AND A1N         B1                                              AND3
AND A1          B1N                                             AND4
AND AON         B0                                              AND5
AND A0          B0N                                             AND6
NOR AND1        AND2                                            NOR1
NOR AND3        AND4                                            NOR2
NOR AND5        AND6                                            NOR3
AND A1N         B1          NOR1                                AND7
AND A1          B1N         NOR1                                AND8
AND AON         B0          NOR1        NOR2                    AND9
AND A0          B0N         NOR1        NOR2                    AND10
AND NOR1        NOR2        NOR3        AGTB                    AND11
AND NOR1        NOR2        NOR3        ALTB                    AND12
AND NOR1        NOR2        NOR3        AEQB                    AND13
OR  AND2        AND8        AND10       AND11                   LFNOT
OR  AND1        AND7        AND9        AND12       AND13       LF
END

MAC SB32
DEFACN          NOR21                                           F0
INV CN                                                          CNNOT
EOR CNNOT       NOR21                                           F0
END
```

```
MAC SB42
DEFANOR11     NOR21     NOR22     CN                F1
AND NOR11     CN                                    A1
AND NOR21     NOR11                                 A2
NOR A1        A2                                    NOR
EOR NOR       NOR22                                 F1
END
```

```
MAC SB52
DEFANOR21     NOR12     NOR11     NOR22     NOR23     F2
DEFACN
AND NOR11     NOR12     CN                          A1
AND NOR11     NOR12     NOR21                       A2
AND NOR12     NOR22                                 A3
NOR A1        A2        A3                          NOR
EOR NOR       NOR23                                 F2
END
```

```
MAC SB63
DEFANOR11     NOR12     NOR13     NOR14     NOR21     F3        CN4
DEFANOR22     NOR23     NOR24     CN
AND NOR11     NOR12     NOR13     CN                  A1
AND NOR11     NOR12     NOR13     NOR21               A2
AND NOR12     NOR13     NOR22                         A3
AND NOR13     NOR23                                   A4
NANDNOR11     NOR12     NOR13     NOR14               PNOT
AND NOR11     NOR12     NOR13     NOR14     NOR21     A6
AND NOR12     NOR13     NOR14     NOR22               A7
AND NOR13     NOR14     NOR23                         A8
AND NOR14     NOR24                                   A9
NOR A1        A2        A3        A4                  NOR1
NOR A6        A7        A8        A9                  GNOT
EOR NOR1      NOR24                                   F3
AND PNOT      GNOT                                    AND1
INV CN                                                CNNOT
AND CNNOT     GNOT                                    AND2
NOR AND1      AND2                                    CN4
END
```

```
MAC SB15
DEFAB          A          SO         SONOT          NOR1       NOR2
INV A                                                AN
INV B                                                BN
AND AN         BN         SO                       A1
AND SONOT      AN         B                        A3
AND AN         BN         SONOT                    A4
AND SO         BN         A                        A5
AND SO         B          AN                       A6
AND B          A          SONOT                    A7
NOR A1         A3                                  NOR1
NOR A4         A5         A6         A7            NOR2
END


MAC SB35
DEFASO         NOR21                               FO
EOR SO         NOR21                               FO
END
```

```
MAC SB65
DEFANOR11     NOR12     NOR13     NOR14     NOR21     F3        CN4
DEFANOR22     NOR23     NOR24     SONOT     SO
AND NOR11     NOR12     NOR13     SONOT               A1
AND NOR11     NOR12     NOR13     NOR21               A2
AND NOR12     NOR13     NOR22                         A3
AND NOR13     NOR23                                   A4
NANDNOR11     NOR12     NOR13     NOR14               PNOT
AND NOR11     NOR12     NOR13     NOR14     NOR21     A6
AND NOR12     NOR13     NOR14     NOR22               A7
AND NOR13     NOR14     NOR23                         A8
AND NOR14     NOR24                                   A9
NOR A1        A2        A3        A4                  NOR1
NOR A6        A7        A8        A9                  GNOT
EOR NOR1      NOR24                                   F3
AND PNOT      GNOT                                    AND1
AND GNOT      SO                                      AND2
NOR AND1      AND2                                    CN4
END
```

```
MAC SB11
DEFAB         A         BN        AN        SO        NOR1      NOR2
DEFAS1
AND AN        BN        SO        S1                  A1
AND SO        BN        A                             A2
AND S1        AN        B                             A3
AND AN        BN                                      A4
AND S1        SO        BN        A                   A5
AND S1        SO        B         AN                  A6
AND B         A                                       A7
NOR A1        A2        A3                            NOR1
NOR A4        A5        A6        A7                  NOR2
END
```

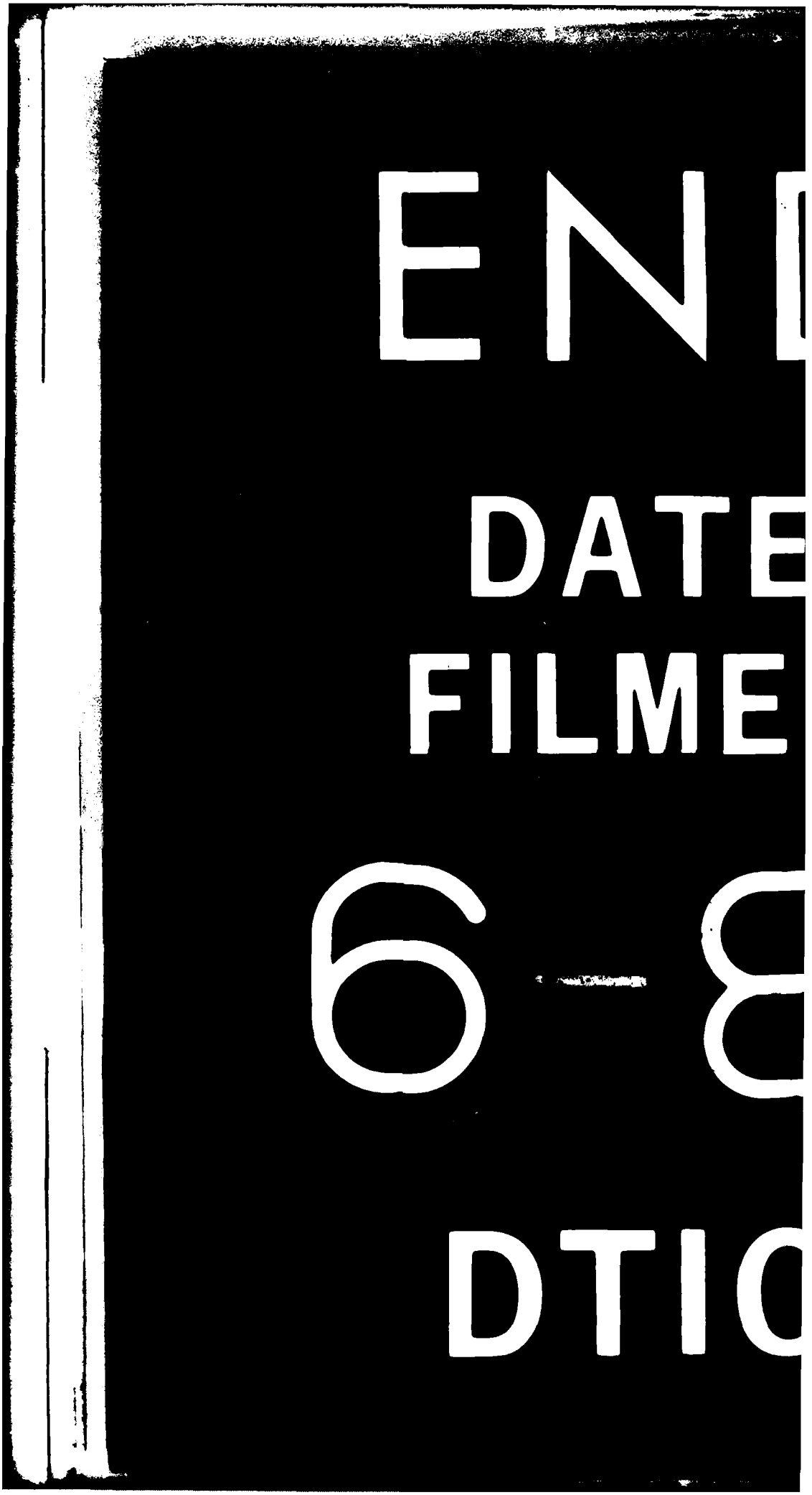
```
MAC SB41
DEFANOR11       F0          NOR22                           F1
AND F0          NOR11                                       A2
NOR NOR11       A2                                          NOR
EOR NOR         NOR22                                       F1
END


MAC SB51
DEFAF0          NOR12       NOR11       NOR22       NOR23   F2
AND NOR11       NOR12                                       A1
AND NOR11       NOR12       F0                              A2
AND NOR12       NOR22                                       A3
NOR A1          A2          A3                              NOR
EOR NOR         NOR23                                       F2
END


MAC SB61
DEFANOR11       NOR12       NOR13       NOR14       F0      F3          CN4
DEFANOR22       NOR23       NOR24
AND NOR11       NOR12       NOR13                           A1
AND NOR11       NOR12       NOR13       F0                  A2
AND NOR12       NOR13       NOR22                           A3
AND NOR13       NOR23                                       A4
NANDNOR11       NOR12       NOR13       NOR14               PNOT
AND NOR11       NOR12       NOR13       NOR14       F0      A6
AND NOR12       NOR13       NOR14       NOR22               A7
AND NOR13       NOR14       NOR23                           A8
AND NOR14       NOR24                                       A9
NOR A1          A2          A3          A4                  NOR1
NOR A6          A7          A8          A9                  GNOT
EOR NOR1        NOR24                                       F3
NANDPNOT        GNOT                                        CN4
END
```

```
MAC SA83
DEFAA       B       C           01      02
DEFA                            03      04
DEFA                            CO
AND C       CO                  01
AND A       CO                  02
AND B       CO                  03
AND A       B       C           04
AND C       A                   AND1
AND C       B                   AND2
AND A       B                   AND3
NOR AND1    AND2    AND3        CO
END

MAC SB83
DEFAA       B                   01      02
DEFA                            03      CO
AND A       CO                  01
AND B       CO                  02
AND A       B                   03
NOR A       B       03          CO
END

MAC SC83
DEFAA       C                   SUM     CO
AND A       C                   SUM
NOR A       C                   CO
END
```

REFERENCES

1. (Author not given), "COMPUTER AIDED DESIGN, Volume III, "LOGIC SIMULATION AND FAULT ANALYSIS PROGRAM, LOGIC 4", USERS MANUAL prepared for Air Force Avionics Laboratory on Contract F33615-76-C-118, April 1977.

*U.S.Government Printing Office: 1980 — 657-084/590


END
DATE
FILME
6-8
DTIC